9th Conference on Algebra and Coalgebra in Computer Science

CALCO 2021, August 31–September 3, 2021, Salzburg, Austria

Edited by

Fabio Gadducci

Alexandra Silva

Editors

Fabio Gadducci
University of Pisa, Italy
fabio.gadducci@unipi.it

Alexandra Silva
Cornell University, USA
University College London, UK
alexandra.silva@gmail.com

ACM Classification 2012
Theory of computation → Models of computation; Theory of computation → Modal and temporal logics; Theory of computation → Algebraic semantics; Theory of computation → Categorical semantics; Theory of computation → Quantum computation theory; Software and its engineering → Context specific languages

ISBN 978-3-95977-212-9

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-212-9.

Publication date
November, 2021

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.CALCO.2021.0

ISBN 978-3-95977-212-9 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Invited Talks

Papers

9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021).
Editors: Fabio Gadducci and Alexandra Silva

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

This volume contains the proceedings of the 9th Conference on Algebra and Coalgebra in Computer Science (CALCO), held both online and at the University of Salzburg, from August 30th to September 3rd, 2021. Previous CALCO editions took place in Swansea (Wales, 2005), Bergen (Norway, 2007), Udine (Italy, 2009), Winchester (UK, 2011), Warsaw (Poland, 2013), Nijmegen (The Netherlands, 2015), Ljubljana (Slovenia, 2017), and London (UK, 2019). This year's edition, following on the tradition started in 2015, was co-located with the conference Mathematical Foundations of Programming Semantics (MFPS).

CALCO is a high-level, bi-annual conference formed by joining CMCS (the International Workshop on Coalgebraic Methods in Computer Science) and WADT (the Workshop on Recent Trends in Algebraic Development Techniques). It provides a forum to present and discuss results of theoretical nature on the mathematics of algebras and coalgebras, the way these results can support methods and techniques for software development, as well as experience reports concerning the transfer of the resulting technologies into industrial practice. Typical topics of interest include

- Models and logics
- Algebraic and coalgebraic semantics
- Methodologies in software and systems engineering
- Specialised models and calculi
- System specification and verification
- Tools supporting algebraic and coalgebraic methods
- String diagrams and network theory
- Quantum computing

At the conference were three invited talks by Holger Giese, Valeria de Paiva, and Viktor Vafeiadis. Additionally, Eugenia Cheng was a joint invited speaker for CALCO and MFPS, and there was a joint special session on Termination Analysis and Synthesis organised by Azadeh Farzan, who delivered an invited tutorial, and with talks by Zac Kincaid and Florian Zuleger. In addition, there were 20 contributed talks, of which 10 were regular papers, 7 (co)algebraic pearls, and 3 early ideas. This volume collects the abstracts of the four invited talks, as well as the peer-reviewed papers.

We are grateful to the Program Committee members for their hard work in reviewing and selecting the papers. They also selected the *Best Paper*, awarded to *Tensor of Quantitative Equational Theories*, authored by Giorgio Bacci, Radu Mardare, Prakash Panangaden, and Gordon Plotkin. The audience instead selected the *Best Talk*, awarded to Ana Sokolova for her presentation of the paper *Nawrotzki's Algorithm for the Countable Splitting Lemma, Constructively*, coauthored with Harald Woracek. Warmest congratulations to the authors!

We would also like to thank the local organisers – Henning Basold, Adriana Pratter, Jurriaan Rot, Sarah Sallinger, Ana Sokolova, and Michael Starzinger – who took on the difficult task of organising a hybrid conference, Thorsten Wißmann, who was publicity chair, and Stefan Milius and Markus Roggenbach, the CALCO steering committee chairs. Our last acknowledgement goes to Michael Wagner and the LIPIcs team, who provided impeccable support in the production of these proceedings.

Fabio Gadducci and Alexandra Silva

9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021).
Editors: Fabio Gadducci and Alexandra Silva
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

Programme Committee

- Zena M. Ariola (University of Oregon)
- Paolo Baldan (University of Padova)
- Rui Soares Barbosa (International Iberian Nanotechnology Laboratory)
- Francisco Durán (University of Málaga)
- Brendan Fong (Massachusetts Institute of Technology)
- Fabrizio Romano Genovese (University of Pisa)
- Jules Hedges (University of Strathclyde, Glasgow)
- Thomas Hildebrandt (IT University of Copenhagen)
- Peter Jipsen (Chapman University)
- Wolfram Kahl (McMaster University)
- Marie Kerjean (CNRS — Laboratoire d'Informatique de Paris Nord)
- Michele Loreti (University of Camerino)
- Sonia Marin (University College London)
- Manuel A. Martins (University of Aveiro)
- Annabelle McIver (Macquarie University)
- Hernan Melgratti (University of Buenos Aires)
- Koko Muroya (RIMS, Kyoto University)
- Elaine Pimentel (Federal University of Rio Grande do Norte)
- Elvinia Riccobene (University of Milan)
- Alex Simpson (University of Ljubljana)
- David I. Spivak (Massachusetts Institute of Technology)
- Christine Tasson (Sorbonne University)
- Tarmo Uustalu (Reykjavik University/Tallinn U. of Technology)
- Maaike Zwart (University of Oxford)
- Rob van Glabbeek (Data61 – CSIRO)

Program Committee Chairs

- Fabio Gadducci (University of Pisa)
- Alexandra Silva (University College London)

Additional reviewers

- Elena Aladova
- Pablo Barenbaum
- Flavien Breuvart
- Thomas Cottrell
- Joerg Endrullis
- Alfredo Roque Freire
- Tobias Fritz
- Richard Garner
- Leandro Gomes
- Clemens Grabmayer

9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021).
Editors: Fabio Gadducci and Alexandra Silva

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Conference Organization

- Adrien Guatto
- Gerco van Heerdt
- Alexander Kurz
- Dylan McDermott
- Alexandre Madeira
- Konstantinos Mamouras
- Paul-André Melliès
- David Jaz Myers
- Andrew Polonsky
- John Power
- Francesco Santini
- Claude Stolze
- Davide Trotta
- Simon Willerton
- Hans Zantema
- Noam Zeilberger

Distributive Laws for Lawvere Theories

Eugenia Cheng ✉
School of the Art Institute, Chicago, IL, USA

Abstract

Distributive laws give a way of combining two algebraic structures expressed as monads; in this work we propose a theory of distributive laws for combining algebraic structures expressed as Lawvere theories. We propose four approaches, involving profunctors, monoidal profunctors, an extension of the free finite-product category 2-monad from Cat to Prof, and factorisation systems respectively. We exhibit comparison functors between CAT and each of these new frameworks to show that the distributive laws between the Lawvere theories correspond in a suitable way to distributive laws between their associated finitary monads. The different but equivalent formulations then provide, between them, a framework conducive to generalisation, but also an explicit description of the composite theories arising from distributive laws.

2012 ACM Subject Classification Theory of computation → Semantics and reasoning

Keywords and phrases Distributive laws, Monads, Lawvere theories

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.1

Category Invited Talk

Related Version The journal paper appeared in *Compositionality*, 2(1)
Full Version: `https://compositionality-journal.org/papers/compositionality-2-1/`

Towards Engineering Smart Cyber-Physical Systems with Graph Transformation Systems

Holger Giese
Hasso Plattner Institute for Digital Engineering, University of Potsdam, Potsdam, Germany

Abstract

A dramatic transformation of our technical world towards smart cyber-physical systems can be currently observed. This transformation results in a networked technical world where besides the embedded systems with their interaction with the physical world the interconnection of these nodes in the cyber world becomes a key element. Furthermore, there is a strong trend towards smart systems where artificial intelligence techniques and in particular machine learning is employed to make software behave accordingly. This raises the question whether our capabilities to model these future embedded systems are ready to tackle the resulting challenges.

In this presentation, we will first discuss how extensions of graph transformation systems can be employed to design and analyse the envisioned future cyber-physical systems with an emphasis on the synergies networking can offer and then characterise which challenges for the design, production, and operation of these systems exist and how they can be tacked with graph transformation systems. We will therefore discuss to what extent our current capabilities in particular concerning engineering with graph transformation systems match these challenges and where substantial improvements for the graph transformation systems have been crucial and will be crucial in the future.

Models are used in classical engineering to plan systems upfront to maximise envisioned properties resp. minimise cost. For smart cyber-physical systems this decoupling of development-time and run-time considerations vanishes, and self-adaptation and runtime models have been advocated as concepts to shift some considerations to run-time. We will review the underlying causes for this shift to run-time, discuss our work with graph transformation systems in this direction, and outline related open challenges and implications for future work for graph transformation systems to engineer smart cyber-physical systems.

2012 ACM Subject Classification Software and its engineering → Software notations and tools

Keywords and phrases Cyber-physical systems, Graph transformation

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.2

Category Invited Talk

9th Conference on Algebra and Coalgebra in Computer Science (CALCO 2021).
Editors: Fabio Gadducci and Alexandra Silva; Article No. 2; pp. 2:1–2:1

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Dialectica Comonads

Valeria de Paiva
Topos Institute, Berkeley, CA, USA

Abstract

Dialectica categories are interesting categorical models of Linear Logic which preserve the differences between linear connectives that the logic is supposed to make, unlike some of the most traditional models like coherence spaces. They arise from the categorical modelling of Gödel's Dialectica interpretation and seem to be having a revival: connections between Dialectica constructions and containers, lenses and polynomials have been described recently in the literature. In this note I will recap the basic Dialectica constructions and then go on to describe the less well-known interplay of comonads, coalgebras and comonoids that characterizes the composite functor standing for the "of course!" operator in dialectica categories. This composition of comonads evokes some work on stateful games by Laird and others, also discussed in the setting of Reddy's system LLMS (Linear Logic Model of State).

2012 ACM Subject Classification Theory of computation → Logic

Keywords and phrases Dialectica categories, Linear logic, Comonads

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.3

Category Invited Talk

Funding *Valeria de Paiva*: Research supported by AFOSR grant FA9550-20-10348.

Acknowledgements I would like to thank Fabio Gadducci and Alexandra Silva for the invitation to speak at CALCO2021. Thinking about the CALCO community and how it influences my work was much more fun than I had expected.

1 Introduction

Monads and comonads, T-algebras and F-coalgebras have been the subject of much investigation within the CALCO (Conference on Algebra and Coalgebra in Computer Science) community, since at least 2005, when the conferences CMCS (Workshop on Coalgebraic Methods in Computer Science) and WADT (Workshop on Algebraic Development Techniques) joined forces to form CALCO. Even earlier on, in 1998, Jacobs, Moss, Reichel and Rutten already wrote in the preface to the first CMCS proceedings that "Coalgebra is beginning to develop into a field of its own, with its own model theory and proof methods (involving bisimulations and invariants)."

The equiparation between algebraic methods and coalgebraic ones is a relatively recent phenomenon, as algebraic methods were much more prevalent in mathematics to begin with. Also in computing the notions of monads and algebras had a head start both with algebraic data types in the eighties and with notions of "computation as monads" in Moggi's and others work in the nineties. Part of the historic phenomenon was indeed that, because categorical dualities are so ubiquitous, many researchers looked at their own tools and results and considered "reversed arrows". This means that the work in coalgebraic methods comes from very different areas and motivations.

Our motivation here is clearly logical, coming from the categorical modelling of (especially intuitionistic) linear and modal logics. We describe quickly Linear Logic, then its models based on Gödel's Dialectica interpretation, then the difficulties of modelling what I take to be

the main innovation of linear logic (the shriek or bang modality $!A$). Finally, we discuss how the models devised (the dialectica spaces) turned out to model other interactions between operators that seem relevant in computing.

2 Linear Logic

Linear logic is a logic proposed by Jean-Yves Girard as a refinement of both classical and intuitionistic logic more than 30 years ago. Linear logic possesses the dualities of classical logic, but also many of the constructive properties of intuitionistic logic. Linear logic lends itself to many different presentations, explanations, and intuitions. Proof-theoretically, it derives from an analysis of classical sequent calculus in which uses of (the structural rules) contraction and weakening are carefully controlled. Operationally, this means that logical deduction is no longer merely about a collection of persistent "truths", but also a way of manipulating resources that cannot always be duplicated or thrown away at will.

In terms of denotational models, linear logic may be seen as refining the interpretation of intuitionistic logic by replacing cartesian (closed) categories by symmetric monoidal (closed) categories. But this is partial, and somewhat misleading because, while models of intuitionistic propositional logic are specific kinds of cartesian closed categories, models of linear logic need to take into account how linear logic relates to intuitionistic logic, so they are better conceived (as has been the case recently) as "symmetric monoidal adjunctions" between a cartesian closed category (modelling the intuitionistic world) and a symmetric monoidal closed category, modelling the linear world, following the work of Benton [5] and others.

This adjunction can be seen as a monoidal comonad (the operator !) in the linear category $\mathcal{L}$ (satisfying some extra properties), or as a monad in the intuitionistic category $\mathcal{C}$, again with extra properties. This monoidal comonad ! behaves very much like a constructive necessity operator $\Box$ in (S4) modal logic, so the whole area of type theory and categorical semantics for Modal Logics can be given another (co)algebraic treatment, different from the usual one, which is based on generalizing its Kripke-like semantics. This kind of type theory and categorical semantics for modal logics can then be used as a blueprint to investigate other modal-like languages, such as hybrid logics, description logics, temporal logics, etc.

A warning is that when we talk about logics, we are considering preferentially constructive or intuitionistic logics. While we would like to sidestep as much as possible philosophical questions about the nature of mathematics, of logic and computation, etc., it seems clear that constructive reasoning principles are safer. If one asks whether "is there an x such that $P(x)$ holds?" most of us will be happier with the answer "yes, x_0", than with an answer of the form "yes, for all x it is not the case that not $P(x)$". We want reasoning to be as precise and safe as possible. Thus, we would like to use constructive reasoning as much as possible. If one needs a classical principle, we hope to be able to flag it, and to know where and why it is needed. Hence, we consider in this note (co)monads and (co)algebras as **constructive modalities** in an extended view of the categorical Curry-Howard correspondence.

2.1 Modalities

Modalities are unary operators over some logic basis. The logical basis can be a classical, an intuitionistic or a linear basis. A logician can think of themselves as exploring the landscape of possible logical systems, trying to answer questions such as: Which modalities? Which basis? Why do I need a modality? How do I choose between modalities?

Axioms:

$$A \to (B \to A)$$
$$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$$
$$A \to (B \to A \wedge B)$$
$$A \wedge B \to A$$
$$A \wedge B \to B$$
$$(A \to C) \to ((B \to C) \to (A \vee B \to C))$$
$$A \to A \vee B$$
$$B \to A \vee B$$
$$\bot \to A$$

Figure 1 Axioms for propositional intuitionistic logic.

To start to answer these questions, we remind the reader that logics can be presented in several different styles. The most traditional formal system to describe a logic is a collection of axioms, in a Hilbert style calculus. For example Figure 1 describes an axiomatization of the intuitionistic propositional calculus. This axiomatization relies on a single rule, Modus Ponens:

$$\frac{\Rightarrow A \to B \qquad \Rightarrow A}{\Rightarrow B}\ \textit{Modus Ponens}$$

(plus substitutions or axiom-schemes)

The main advantage of axiomatic systems are that it is easier to prove theorems *about* a logic described using them. The main problem with axiomatic systems is that it is difficult to prove theorems *within* the logic, but also that there is no canonical system of axioms. The axioms can be true for very obscure reasons. There are several choices of axioms and no obvious criterion to choose between the possibilities.

Gentzen systems of sequent calculus and natural deduction are much more reliable. Sequent calculus systems consist of structural and logical rules. For each logical connective we have left and right rules. The main advantage of sequent calculi is that it is easier to find proofs in the system, which facilitates automated deduction. The system is very symmetrical, and the geometry of derivations helps to get theorems about the working of the system itself.

Natural Deduction is the other Gentzen system for proof construction and natural deduction formulations of logic are usually preferred by logicians interested in proofs. These model better how humans reason, hence the adjective "natural". Many proofs assistants use Natural Deduction, either in Prawitz or in Martin-Löf style, as their formalism to describe logics.

Looking at presentations of the sequent calculus we can see the cleverness behind Girard's Linear Logic. Girard removed the structural rules of contraction and weakening from usual rules of logic, so the logic becomes resource aware. One cannot discard premises not used in a proof (weakening), one cannot duplicate a premise A that we used once (contraction).

$$\frac{\Gamma \Rightarrow C}{\Gamma,\, A \Rightarrow C}\ \textit{Weakening}$$

$$\frac{\Gamma,\, A,\, A \Rightarrow C}{\Gamma,\, A \Rightarrow C}\ \textit{Contraction}$$

However, if you simply remove these two structural rules, the logic is too weak. To get back the expressive power of usual logic, Girard uses a modality, written as ! and called "of course!". This works mostly via the translation

$$A \to B = (!A) \multimap B.$$

Contraction and weakening are available only for $!A$ formulas/objects. The operator ! (shriek or bang) is a unary operator over a linear basis whose rules correspond to the $\Box$ (or necessity) S4 modality. It was realized very early on that this modality should be modelled categorically by a comonad.

Why? Which kind of comonad? Before facing these questions, we expand our description of a propositional linear basis and its categorical modelling.

2.2 The challenges of modeling Linear Logic

When modelling of Linear Logic is discussed the first task is to explain the difference between a resource-aware implication and an usual one. A traditional implication satisfies:

$A, A \to B \ \vdash\ B$
$A, A \to B \ \vdash\ A \wedge B$

Thus one can use Modus Ponens (MP) to obtain B, from A and $A \to B$, but we still have A after completing the inference. Meanwhile a linear implication satisfies instead:

$A, A \multimap B \ \vdash B$
$A, A \multimap B \ \nvdash A \otimes B$

As before we can deduce B from A and $A \multimap B$, but A is then gone, we cannot produce it again, from nowhere. Similarly, the traditional conjunction $A \wedge B$ satisfies $A \wedge B \vdash B$ (or A) while a linear conjunction does not entail its conjuncts $A \otimes B \ \nvdash A$. We cannot project A (or B) from a linear conjunction.

When we add the modality ! we can of course re-use formulae and discard them, if they have the right type.

Of course:	$!A \ \vdash\ !A \otimes !A$	Re-use
	$!A \otimes B \ \vdash\ I \otimes B \cong B$	Discard

The traditional categorical modeling of intuitionistic logic, associates a formula A to an object A of an appropriate (i.e. a cartesian closed) category $\mathbb{C}$. The intuitionistic conjunction $A \wedge B$ is associated to a cartesian product $A \times B$ and the intuitionistic implication $A \to B$ is associated to a function space B^A (the homset in $\mathbb{C}$ of functions from A to B).

The usual conjunction are modelled by real cartesian products, so we have projections $(A \times B \to A, B)$, and diagonals $(A \to A \times A)$, which correspond to deletion and duplication of resources. This is not a linear setting. To cope with the removal of the structural rules of weakening and contraction, the structure needs to be relaxed from a cartesian closed category to a monoidal closed category, with tensors and internal-homs. This move is not problematic, as category theorists had the mathematics done several decades before they were needed. We may also need to have coproducts and products to model the linear logic additives (written as $\oplus$ and &), if these are desired. But these structures were also part of the category theorists vocabulary and their adaptation to the then new setting was not problematic.

However, the main challenge of modelling linear logic was really the operator $!A$. Let us recap its rules:

$$\frac{!\Gamma \vdash B}{!\Gamma \vdash !B} \qquad \frac{\Gamma \vdash B}{\Gamma, !A \vdash B} \qquad \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \qquad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}$$

If in the first rule above we substitute $!\Gamma$ for $!A$ and B for $!A$, we end up with

$$\frac{!A \vdash !A}{!A \vdash !!A}$$

which shows that we need to have a morphism in the category with shape $\delta\colon !A \to !!A$, as any category has the identity $id_{!A}\colon !A \to !A$. Similarly, for the other rules below:

$$\frac{\Gamma = !A, B = !A}{\delta\colon !A \to !!A} \qquad \frac{\Gamma = \emptyset, B = I}{\mathsf{er}\colon !\mathsf{A} \to \mathsf{I}} \qquad \frac{\Gamma = \emptyset, B = A}{\mathsf{der}\colon !\mathsf{A} \to \mathsf{A}} \qquad \frac{\Gamma = \emptyset, B = !A \otimes !A}{\mathsf{copy}\colon !\mathsf{A} \to !\mathsf{A} \otimes !\mathsf{A}}$$

Thus we see that, for each object of the category $\mathbb{C}$ of the form $!A$ we must have a collection of natural morphisms:

$$!A \xrightarrow{\mathsf{prom}} !!A$$

$$!A \xrightarrow{\mathsf{counit}} I$$

$$!A \xrightarrow{\mathsf{der}} A$$

$$!A \xrightarrow{\mathsf{copy}} !A \otimes !A$$

Then we infer that $!A$ needs to be a commutative comonoid with respect to the tensor product. But also that it should be a coalgebra for the monoidal comonad "!". The comonoid and the coalgebra structures need to interact in a consistent and coherent way, which makes many equations required. Which is the best way to describe these interactions?

3 Dialectica spaces

Historically Seely [28], Lafont [17] and de Paiva [9] presented their categorical models more or less at the same time, around 1987. But de Paiva investigated a particular, concrete case of Seely's generic idea. Her original model also happens to be an instance of Lafont's cofree model.

Seely isomorphisms, already present in Girard's original paper, mix multiplicative and additive categorical structures.

$$!(A \& B) \cong !A \otimes !B \text{ and } !1 \cong I$$

Work by Benton et al [4] and Bierman [6] came with a solution of model (for categories without products), but the solution involved too many "commuting conversions". Benton's LNL system [5] and Barber's DILL "Dual Intuitionistic and Linear Logic"[2], as well as Maietti et al [23] "Relating Models of Intuitionistic Linear Logic" and Hyland and Schalk's [20] glueing models have a working theory of what models should look like. A more complete discussion, with full proofs, but from a very different perspective, can be found in Melliès [1].

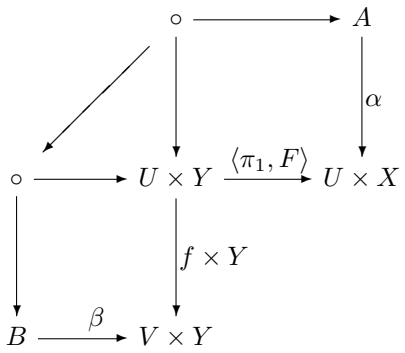
3.1 Dialectica Spaces

The Dialectica constructions give us a precision-based model of Linear Logic. All connectives correspond to distinct structures in the categories. We describe two families of models: the original Dialectica categories $Dial_2(\mathbb{C})$ model intuitionistic linear logic, while the Dialectica-like construction, suggested by Girard, the categories $DDial_2(\mathbb{C})$, model classical or full intuitionistic linear logic. We use the expression *dialectica spaces* to refer to both constructions.

The difference between these constructions is mostly on morphisms, objects are the same, i.e. triples $(U, X, \alpha : A \hookrightarrow U \times X)$ where U, X are objects of $\mathbb{C}$, a cartesian closed category and α a relation between the objects U, X . However, different notions of morphism will produce very different structures in the resulting categories. However, both constructions produce dialectica categories $Dial_2(\mathbb{C})$ and $DDial_2(\mathbb{C})$ which are symmetric monoidal closed categories with products, as we recall below.

The first construction, $Dial_2(\mathbb{C})$, comes from the internalization of the functionals in Gödel's Dialectica interpretation. Given a cartesian closed category $\mathbb{C}$, we construct the category $Dial_2(\mathbb{C})$ as follows. The objects of $Dial_2(\mathbb{C})$ are relations between U and X, so an object in $Dial_2(\mathbb{C})$ is a monic $A \overset{\alpha}{\hookrightarrow} U \times X$ written as $(U \overset{\alpha}{\leftarrow\!\!+} X)$.

A map in $\mathsf{Dial}_2(C)$ between objects $A = (U, X, \alpha)$ and $B = (V, Y, \beta)$ is a pair of maps in C, $(f\colon U \to V, F\colon U \times Y \to X)$ satisfying the pullback condition below. We can think of this condition in **Set** as saying if $u\alpha F(u, y)$ then $f(u)\beta y$. Thus



This is where the mathematics gets somewhat involved, as calculating pullbacks can be tiresome. One has to prove that the definitions make sense and moreover that they work. That is, the category $Dial_2(\mathbb{C})$ with the structure described below is a symmetric monoidal closed category with products. The tensor product, the internal-hom and the categorical product in $Dial_2(\mathbb{C})$ are given as follows:

$$A \oslash B = (U \times V \overset{\alpha \oslash \beta}{\leftarrow\!\!+} X \times Y)$$

$$A \multimap B = (V^U \times X^{U \times Y} \overset{\alpha \multimap \beta}{\leftarrow\!\!+} U \times Y)$$

$$A \& B = (U \times V \overset{\alpha\beta}{\leftarrow\!\!+} X + Y)$$

The category has only *weak coproducts* and a weak initial object. The full calculations and properties needed to show that this structure works as required can be found in the technical report [14].

▶ **Theorem 1** (de Paiva 1987). *The category $Dial_2(\mathbb{C})$ has a symmetric monoidal closed structure (and products and weak coproducts), that makes it a model of (exponential-free)* **intuitionistic** *linear logic.*

The category $Dial_2(\mathbb{C})$ is a good model for Linear Logic, since it has the structure needed to model intuitionistic linear logic, including a highly non-trivial and very well-behaved modality !. The model was an independent confirmation of the worth of Linear Logic, since it is based on Gödel's Dialectica interpretation, an interesting, useful and still a bit mysterious tool in proof theory [21]. However, it was not clear if there was an obvious way of transforming this model into a model of Classical Linear Logic.

We can reformulate $Dial_2(\mathbb{C})$ a little: Say we consider the category $\mathbb{C}$ as **Set**, objects are maps $U \times X \xrightarrow{\alpha} 2$, morphisms are pairs of maps $f\colon U \to V$, $F\colon U \times Y \to X$ such that in the diagram

$$\begin{array}{ccc} U \times Y & \xrightarrow{\langle \pi_1,\, F\rangle} & U \times X \\ {\scriptstyle f\times id_Y}\downarrow & & \downarrow{\scriptstyle \alpha} \\ V \times Y & \xrightarrow[\beta]{} & 2 \end{array}$$

we have $\alpha(u,\, F(u,y)) \leq \beta(fu, y)$. The point of this formulation is to compare the categories with other proposed models of linear logic like *Chu spaces* [24].

Following a suggestion of Girard a substantial modification can be made to the category $Dial_2(\mathbb{C})$ to obtain a model of Classical Linear Logic. Actually, in the general case, the category we obtain is a model of Full Intuitionistic Linear Logic [19], but if desired we can restrict ourselves to a model of classical linear logic.

Again we assume that the category $\mathbb{C}$ is a cartesian closed category. The objects of the category $DDial_2(\mathbb{C})$ are relations between U and X in $\mathbb{C}$, as before. However, the maps of $DDial_2(\mathbb{C})$ are now simply inverse pairs of maps of $\mathbb{C}$, $f\colon U \to V$, $F\colon Y \to X$ such that $u\alpha F(y) \Rightarrow f(u)\beta y$. Thus the only difference is that in $Dial_2(\mathbb{C})$, the "counterexample" map F has domain $U \times Y$, while in $DDial_2(\mathbb{C})$, F has domain simply Y.

This simplification of the notion of morphism gives us more structure still to describe. But the work with the categories $Dial_2(\mathbb{C})$ helps to make intuitions clearer. Summarizing the structure in $DDial_2(\mathbb{C})$ is given by:

$$\begin{aligned} A \multimap B &= (V^U \times X^Y \xleftarrow[+]{\alpha \multimap \beta} U \times Y) \\ A \otimes B &= (U \times V \xleftarrow[+]{\alpha \otimes \beta} X^V \times Y^U) \\ A \parr B &= (V^X \times U^Y \xleftarrow[+]{\alpha \parr \beta} X \times Y) \\ A \,\&\, B &= (U \times V \xleftarrow[+]{\alpha \,\cdot\, \beta} X + Y) \\ A \oplus B &= (U + V \xleftarrow[+]{\alpha \,\cdot\, \beta} X \times Y) \end{aligned}$$

For $Dial_2(\mathbb{C})$ the tensor product was easy to obtain, and we worked what the internal-hom should be, to obtain the adjunction we wanted. For $DDial_2(\mathbb{C})$ the opposite process occurs: we now know the shape of the internal-hom and have to guess an appropriate tensor-product to obtain the monoidal closure. The guessing is shown to work, by proving the adjunction that says that the category is symmetric monoidal closed.

▶ **Theorem 2** (de Paiva 1988). *The category $DDial_2(\mathbb{C})$ has a symmetric monoidal closed structure (and products, coproducts and units), that makes it a model of (exponential-free)* **classical** *linear logic.*

Products and coproducts work as expected. Also the units now work as expected, we have four distinct units and calculations can be found, as before, in the technical report:

$$\begin{aligned} I &= (1 \xleftarrow{+}{}^{i} 1) \\ \bot &= (1 \xleftarrow{+}{}^{e} 1) \\ 1 &= (1 \xleftarrow{+}{}^{\cdot} 0) \\ 0 &= (0 \xleftarrow{+}{}^{\cdot} 1) \end{aligned}$$

Having prepared the linear logic basis (twice!), we now go back to the more interesting issue of the modalities, that is our comonads and their coalgebras and comonoids.

4 Dialectica Comonads

We want *all* the expressive power of traditional logic. For that Linear Logic introduces the modalities ! and ?, which are dual to each other. The intuition for these modalities is well-known: $!A$ means we can use A as many times as we want and, dually, the operator $?A$ means we can create as many A's as desired.

In this section, we first define a modality ! in $Dial_2(\mathbb{C})$ and then, to define the operator "!" for $DDial_2(\mathbb{C})$, we have to compose comonads. We abstract a few lessons from this process.

4.1 A cofree !-comonad

Surprisingly enough the ! operator in $Dial_2(\mathbb{C})$ is a *cofree* comonad. There are many useful and well-known monads in the literature like exceptions, side-effects, powerset, continuations, etc... but there are fewer well-known and loved comonads. Since we want a comonad that creates comonoids with respect to the tensor product in the category $Dial_2(\mathbb{C})$, we want an object $!A$ such that it has a diagonal like function to $!A \otimes !A$. Now whatever the ! operator is, the tensor product $!A \otimes !A$ needs to have something like $U \times U$ as its left-handed side. This is easy, as $\mathbb{C}$ is cartesian closed, we can use its diagonal. But we need to also construct a map of shape $U \times X \times X \to X$, so it would be nice to have a monoid structure in X. It turns out that simply asking for free commutative monoids in $\mathbb{C}$ is enough.

This corresponds to the fact that our category $Dial_2(\mathbb{C})$ has objects with dual aspects to it. The left-handed object U, in the triple (U, X, α), tends to behave as expected in a cartesian closed category, that is, it works like a verifier of assertions. Meanwhile, the right-handed side object, X, tends to behave like a producer of counter-examples to the original claim made by the whole object $A = (U, X, \alpha)$ that, internally, there exists an u in U, such that for all x in X, $\alpha(u, x)$ holds.

Thus in our case, to obtain a comonoid in $Dial_2(\mathbb{C})$ we simply need an object that is a monoid in the second coordinate, this not only defines a comonad, but this comonad is a cofree one.

▶ **Definition 3.** *Define a comonad in $Dial_2(\mathbb{C})$ by saying $!(U, X, \alpha) = (U, X^*, \alpha^*)$, where X^* is the free commutative monoid in $\mathbb{C}$ and the new relation α^* is simply a repetition of the α relation, as many times as necessary for the sequences in X^*.*

Then we prove that this is the cofree comonad in $Dial_2(\mathbb{C})$, that is, that there is an adjunction between the forgetful functor from the category of commutative comonoids in $Dial_2(\mathbb{C})$ and $Dial_2(\mathbb{C})$ itself.

▶ **Theorem 4** (de Paiva 1987). *The monoidal comonad ! in $Dial_2(\mathbb{C})$ models linear logic modalities and recovers intuitionistic propositional logic.*

To prove this, we transform it into

▶ **Lemma 5.** *The co-Kleisli category associated with the (symmetric monoidal) comonad ! on $Dial_2(\mathbb{C})$ is cartesian closed.*

Proof. To show cartesian closedness we need to show:

$$Hom_{Kl!}(A\&B, C) \cong Hom_{Kl!}(A, [B, C]_{Kl!})$$

The proof is then a series of equivalences that were proved previously:
$Hom_{Kl!}(A\&B, C) \cong Hom_{Dial_2(\mathbb{C})}(!(A\&B), C) \cong Hom_{Dial_2(\mathbb{C})}(!A \otimes !B, C)$
$\cong Hom_{Dial_2(\mathbb{C})}(!A, [!B, C]_{Dial_2(\mathbb{C})}) \cong Hom_{Kl!}(A, [!B, C]_{Dial_2(\mathbb{C})}) \cong Hom_{Kl!}(A, [B, C]_{Kl!})$
This is described with details in [28] and section 2.5 of [14]. ◀

This was the first cofree comonad model of linear logic, not purely syntactic, that is, not simply of the form this is what we want to be the case. It has inspired the comonad in Chu spaces, as described by Lafont and Streicher [22].

The operator ! is a monoidal comonad, however, the significance of *monoidal* was not realized to begin with. That means that there is a natural transformation

$$m_{(-,-)} : !A \otimes !B \to !(A \otimes B) \text{ and a morphism } M_I : I \to !I$$

satisfying many commutative diagrams. The operator ! induces a commutative comonoid structure on the object $!A$, but $!A$ also has naturally a coalgebra structure induced by the comonad !. The comonoid and coalgebra structures must interact in a nice way and how to describe this nice way is up to controversy.

4.2 Exponential comonads for Dialectica-like constructions

The Dialectica-like category $DDial_2(\mathbb{C})$ [15] is a simpler category than $Dial_2(\mathbb{C})$. Since we simplified the category, does it mean we simplified the exponential comonad used to embed it into a traditional cartesian universe? Unfortunately, the answer is no. The previous comonad we had for $!A$ in $Dial_2(\mathbb{C})$ does not work for $DDial_2(\mathbb{C})$: we need commutative comonoids and coalgebras with respect to the new tensor product, which is much more complicated than the tensor product of $Dial_2(\mathbb{C})$.

Tensor products should be adjoint to internal-homs, which internalize the morphisms of the category under discussion. The category $DDial_2(\mathbb{C})$ has simpler morphisms than $Dial_2(\mathbb{C})$: the internal-hom of $DDial_2(\mathbb{C})$ is given by

$$A \multimap B \quad = \quad (V^U \times X^Y \xleftarrow{\alpha \multimap \beta} U \times Y)$$

where counterexample morphisms are simply from Y to X, as explained before. We can reverse engineer a tensor product that works well with the simplified internal-hom of $DDial_2(\mathbb{C})$ above. This tensor product is given by

$$(U, X, \alpha) \otimes (V, Y, \beta) = (U \times V, X^V \times Y^U, \alpha \otimes \beta).$$

But this is more complicated than simply putting relations side-by-side as we had for the category $Dial_2(\mathbb{C})$.

Given that the the previous comonad ! worked so well in the category $Dial_2(\mathbb{C})$ and that it is still a comonad in $DDial_2(\mathbb{C})$, we can call it $\mathtt{bang}_1$ and use it to deal with the comonoid structure. Note that in the previous definition we never used the dependency in U of the morphisms considered.

▶ **Definition 6.** *Define a functor in $DDial_2(\mathbb{C})$ by saying $\mathtt{bang}_1(U, X, \alpha) = (U, X^*, \alpha^*)$, where X^* is the free commutative monoid in $\mathbb{C}$ and the new relation α^* is simply a repetition of the α relation, as many times as necessary for the sequences in X^*.*

To have coalgebras we need a different comonad.

▶ **Definition 7.** *Define a functor in $DDial_2(\mathbb{C})$ by saying $\mathtt{bang}_2(U, X, \alpha) = (U, X^U, \alpha^U)$, where $(-)^U : \mathbb{C} \to \mathbb{C}$ is the corresponding monad, for a given U, in $\mathbb{C}$.*

This induces a comonad in $\mathsf{DDial}_2(C)$ but $\mathtt{bang}_2$ isn't enough!

The good news is that we can compose these comonads, unlike most comonads. We just need a distributive law, which had been already described by Beck [3]. (The suggestion of using a composite of comonads came from Th. Coquand, and it was much appreciated.)

Take $\mathtt{bang}(U, X, \alpha) = (U, (X^*)^U, (\alpha*)^U)$, where $(-)^*$ is the free commutative monoid in $\mathbb{C}$ and $(-)^U : \mathbb{C} \to \mathbb{C}$ is a monad in $\mathbb{C}$ that induces a comonad in $\mathsf{DDial}_2(C)$. Most of the calculations are in [14] and [15], except the ones for monoidicity of the (co)monads involved.

▶ **Proposition 8.** *There is a comonad* bang *in $\mathsf{DDial}_2(\mathbb{C})$, a composite of two comonads, which models the modality ! in Linear Logic.*

▶ **Theorem 9** (Linear-Non-Linear version)**.** *There is a* **monoidal** *adjunction between $\mathsf{DDial}_2(\mathbb{C})$ and its coKleisli category for the composite monoidal comonad* bang *described above.*

A comonad that is a composite of two comonads is harder to deal with than the situation we had with $\mathsf{Dial}_2(\mathbb{C})$. The fact that this comonad is obtained from a composite monad in $\mathbb{C}$ and that we need to use two distributive laws (one for the monads in $\mathbb{C}$, one for the comonads in $\mathsf{DDial}_2(\mathbb{C})$) makes the whole enterprise somewhat convoluted. One might be led to think that this is a "hack", without further significance. The calculations get bigger and cumbersome, but do not present intrinsic difficulties. They require a small amount of Street's theory of monads [31] suitably dualized in places.

But something more interesting is happening here: the comonads come with their own appropriate tensor products, as they need to be monoidal, for logical reasons. The comonad ! in $\mathsf{Dial}_2(\mathbb{C})$ is very well-behaved, it is cofree. The composite comonad bang in $\mathsf{DDial}_2(\mathbb{C})$ seems very *ad hoc*, it is clearly not cofree, but it is monoidal and this is enough for the modelling role we wanted it for. The calculations showing that all the commutative diagrams really commute are easy, but there are many of them and they are long. Computer systems can help with them, e.g. the Agda report from Eades in `https://github.com/heades/cut-fill-agda` for the paper [16].

These constructions look ad hoc, but they have been used repeatedly and reappear in many other projects, starting with [8] and [11]. We describe this use in the next section. We also draw the reader's attention to the formal similarity of these notions of multiple tensors with associated comonads to the work of Retoré in [26] and of Laird anf Churchill on sequentiality [7].

5 Modelling State?

The concept of state-manipulation plays an important role in computation. There are many suggestions of how to formalise this concept, especially using type systems derived from Girard's Linear Logic. Linear Logic was long considered a candidate logic for modelling "state" as usually considered in programming languages. Reddy introduced his system LLMS (Linear Logic Model of State) in 1993 [25] to address the issue of state in functional and imperative programming languages. As he says "This work addresses twin issues: how to incorporate state manipulation in functional programming languages, and how to describe the semantics of higher-order imperative programming languages."

Reddy's system is an extended intuitionistic linear calculus with extra connectives for modelling state manipulation. The system LLMS adds two extra connectives to those of Intuitionistic Linear Logic: first, a sequencing binary operator $\triangleright$, called "before", which is a non-commutative tensor product and a "regenerative" storage operator †, a modality associated with "before", in a way that parallels the relationship between tensor and the exponential !.

Discussing the system, Reddy explains the connective 'before' $\triangleright$ as the denotation of sequential composition of components. Meanwhile the "regenerative" storage operator † allows him to build sequentially reusable storage objects. The approach is shown to work by embbeding a higher-order Algol-like language in the system LLMS.

If "before" captures composition with possible left to right communication, it stands to reason that it should be non-commutative. But we can consider a commutative variant of the connective, obtaining a non-ordered combination of elements. This was the work in [8]. In that work non-ordered combination does not mean communication both ways. Instead it represents concurrent execution without interaction or synchronization, which was a suggestion of Haeusler.

The work with Correa and Haeusler [8] follows exactly Reddy's description, which in its turn is based on Retoré Pomset logic [26]. Pomset logic is not an easy system, and it is particularly difficult, if "proof-nets" are not your area of expertise. A context Γ is characterized as a pomset (a partially ordered multiset) $(|\Gamma|, \leq_\Gamma)$, where $|\Gamma|$ is the set of formula occurrences in Γ, and $\leq_\Gamma$ is a partial order on the context $|\Gamma|$. It turns out that the commutative version of LLMS, LLMS_c, can be soundly modeled by the dialectica category $\mathsf{DDial}_2(\mathbb{C})$ and LLMS itself can be modelled by a suitable (non-commutative) variant of the dialectica construction [11]. This non-commutative variant of the dialectica construction had first been considered as a model for the Lambek calculus in [10].

The intrinsic order of the pomset context makes proofs much harder and leads into issues discussed by the community interested in Deep Inference. Many questions are still open, as discussed recently by Retoré [27] and Slavnov [29]. But some answers can be provided.

▶ **Theorem 10** (cut elimination). *If a sequent is provable in the commutative system $LLMS_c$, then it is provable in $LLMS_c$ without any application of the cut rule.*

Cut elimination together with the theorem below, are the main results of [8], together with the interpretation in terms of CSP processes of formulae of the system.

▶ **Theorem 11** (Corrêa, Haeusler, de Paiva, 1996). *The dialectica categories $\mathsf{DDial}_2(\mathbb{C})$ are a sound model of $LLMS_c$.*

The fact that the ad hoc construction of the dialectica-like modality was recreated, independently, in the totally unrelated field of managing state in functional and imperative programming languages, makes one wonder. The construction seems to be robust. As it is shown in [11] we also have

▶ **Theorem 12** (Tucker 1998). *There exists a small variant of the dialectica categories* $\mathsf{DDial}_2(\mathbb{C})$*, where the original, non-commutative system LLMS can be soundly modeled.*

6 Conclusions

We discussed some stories of monads, comonads, algebras and coalgebras that appear in work related to logic systems oriented towards computing. My main examples have to do with Linear Logic, so I called these "Linear Modalities" as (co)monads behave like S4 modalities when considered as components of proof systems over a linear basis. I hope every one can see how Dialectica spaces of several kinds introduce different (co)monads useful to provide models of several possible extensions of linear logic.

I have not discussed much other constructive modal logics, which is also work that goes under the heading of (co)monads and (co)algebras for Computer Science. Constructive modal logics are interesting for programmers, logicians, mathematicians and philosophers who know about them under different names and for different applications. It is a shame that these communities do not talk much to each other. The consortium of researchers united under the title "Intuitionistic Modal Logics and Applications" (IMLA) [30, 13, 12] has been trying to change this since 1999. It would be good if this work could be continued and improved. There is plenty of future work along these lines, producing categorical models for systems with linear modalities of different styles.

I have started the talk, that is now written as this note, with a quote from Martin Hyland in "Proof Theory in the Abstract" [18]. He says

> Elegant mathematics will of itself tell a tale, and one with the merit of simplicity. This may carry philosophical weight. But that cannot be guaranteed: in the end one cannot escape the need to form a judgement of significance.

Working within the confines of categorical versions of the Curry-Howard correspondence has been a guiding principle for me, others can find their own judgements of significance. There is plenty of interesting work to do.

References

1. Paul André Melliès. Categorical semantics of linear logic. In *In: Interactive Models of Computation and Program Behaviour, Panoramas et Synthèses 27, Société Mathématique de France 1–196*, 2009.
2. Andrew Barber and Gordon Plotkin. *Dual intuitionistic linear logic.* University of Edinburgh, Department of Computer Science, Laboratory for . . . , 1996.
3. Jon Beck. Distributive laws. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, pages 119–140, Berlin, Heidelberg, 1969. Springer Berlin Heidelberg.
4. Nick Benton, Gavin Bierman, Valeria De Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *International Conference on Typed Lambda Calculi and Applications*, pages 75–90. Springer, 1993.
5. P. N. Benton. A mixed linear and non-linear logic: Proofs, terms and models. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic*, pages 121–135, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.
6. Gavin M Bierman. What is a categorical model of intuitionistic linear logic? In *International Conference on Typed Lambda Calculi and Applications*, pages 78–93. Springer, 1995.
7. Martin Churchill and James Laird. A logic of sequentiality. In *International Workshop on Computer Science Logic*, pages 215–229. Springer, 2010.

8 Marcelo da S Corrêa, Edward H Haeusler, and Valeria CV de Paiva. A dialectica model of state. In *CATS'96, Computing: The Australian Theory Symposium Proceedings*, pages 11–12, 1996.

9 Valeria de Paiva. The dialectica categories. *Categories in Computer Science and Logic*, 92:47–62, 1989.

10 Valeria De Paiva. A dialectica model of the lambek calculus. In *AMSTERDAM COLLOQUIUM, 1991*. Citeseer, 1991.

11 Valeria De Paiva. Linear logic model of state revisited. *Logic Journal of the IGPL*, 22(5):791–804, 2014.

12 Valeria de Paiva, Rajeev Goré, and Michael Mendler. Modalities in constructive logics and type theories, 2004.

13 Valeria de Paiva and Brigitte Pientka. Intuitionistic modal logic and applications (IMLA 2008). *Information and computation (Print)*, 209(12), 2011.

14 Valeria Correa Vaz de Paiva. The dialectica categories. Technical report, University of Cambridge, Computer Laboratory, Technical Report 213, 1991. URL: `https://128.232.0.20/techreports/UCAM-CL-TR-213.pdf`.

15 Valeria CV De Paiva. A dialectica-like model of linear logic. In *Category Theory and Computer Science*, pages 341–356. Springer, 1989.

16 Harley Eades and Valeria de Paiva. Multiple conclusion linear logic: Cut elimination and more. In Sergei Artemov and Anil Nerode, editors, *Logical Foundations of Computer Science*, pages 90–105, Cham, 2016. Springer International Publishing.

17 Jean-Yves Girard and Yves Lafont. Linear logic and lazy computation. In *International Joint Conference on Theory and Practice of Software Development*, pages 52–66. Springer, 1987.

18 J.M.E. Hyland. Proof theory in the abstract. *Annals of Pure and Applied Logic*, 114(1):43–78, 2002. Troelstra Festschrift. `doi:10.1016/S0168-0072(01)00075-6`.

19 Martin Hyland and Valeria De Paiva. Full intuitionistic linear logic. *Annals of Pure and Applied Logic*, 64(3):273–291, 1993.

20 Martin Hyland and Andrea Schalk. Glueing and orthogonality for models of linear logic. *Theoretical computer science*, 294(1-2):183–231, 2003.

21 Ulrich Kohlenbach. *Applied proof theory: proof interpretations and their use in mathematics*. Springer Science & Business Media, 2008.

22 Yves Lafont and Thomas Streicher. Games semantics for linear logic. In *Proceedings 1991 Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 43–44. IEEE Computer Society, 1991.

23 Maria Emilia Maietti, Paola Maneggia, Valeria De Paiva, and Eike Ritter. Relating categorical semantics for intuitionistic linear logic. *Applied categorical structures*, 13(1):1–36, 2005.

24 Vaughan Pratt. Chu spaces. *School on Category Theory and Applications (Coimbra, 1999)*, 21:39–100, 1999.

25 Uday S. Reddy. A linear logic model of state. *Manuscript*, 1993.

26 Christian Retoré. Pomset logic: A non-commutative extension of classical linear logic. In Philippe de Groote and J. Roger Hindley, editors, *Typed Lambda Calculi and Applications*, pages 300–318, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

27 Christian Retoré. Pomset logic: a logical and grammatical alternative to the lambek calculus, 2020. `arXiv:2001.02155`.

28 R.A.G. Seely. Linear logic, *-autonomous categories and cofree coalgebras. In *Categories in Computer Science and Logic*, pages 371–382. American Mathematical Society, 1989.

29 Sergey Slavnov. On noncommutative extensions of linear logic. *arXiv preprint arXiv:1703.10092*, 2017.

30 Charles Stewart, Valeria de Paiva, and Natasha Alechina. Intuitionistic modal logic: a 15-year retrospective. *Journal of Logic and Computation*, 2018.

31 Ross Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2(2):149–168, 1972. `doi:10.1016/0022-4049(72)90019-9`.

The Challenges of Weak Persistency

Viktor Vafeiadis
MPI-SWS, Kaiserslautern, Germany

Abstract

Non-volatile memory (NVM) is a new hardware technology that provides durable storage at performance similar to that of plain volatile RAM. As such, there is a lot of interest in exploiting this technology to improve the performance of existing disk-bound applications and to find new applications for it. Nevertheless, developing correct programs that interact with non-volatile memory is by no means easy, since mainstream architectures provide rather weak persistency semantics and rather low-level and expensive mechanisms in order to avoid weak behaviors. This creates many opportunities for researchers in programming language semantics, logic, and verification to develop techniques to assist programmers writing NVM programs. This short paper and the associated talk outline the challenges caused by NVM and the research opportunities for PL researchers.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Weak Persistency, Non-Volatile Memory

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.4

Category Invited Talk

Funding My ongoing work on persistency is partly funded from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101003349, project name PERSIST).

1 Introduction

Until recently, computer storage came in two flavors: (1) volatile storage, such as DRAM, that enables fast byte-sized access but loses its contents upon a power outage, and (2) durable storage, such as hard drives, that preserves its contents upon a power failure but provides only slow block-sized access to data. Due to these characteristics of computer storage, software applications use different mechanisms to access memory and disks.

All this is about to change with the adoption of non-volatile memory (NVM), a recent technology that provides durable storage with performance similar to that of volatile memory. NVM is plugged to the memory bus and can be accessed in byte-sized chunks with latency and throughput slightly worse than those of DRAM (within an order of magnitude). It is thus expected that NVM will soon supplant or even replace volatile memory.

2 Results and Open Questions

Non-volatile memory raises a number of questions relevant for researchers in programming language semantics and verification. Although recent research in this area has scratched the surface of these questions providing some first answers to them, much more research is needed to provide more thorough answers. In the following, I summarize these questions and partial answers in four groups.

First, what *semantics* do programs interacting with non-volatile memory have? Among the multiple ways of defining semantics, to date, only operational and declarative/axiomatic approaches have been applied to model the semantics of persistent programs. Specifically, Raad et al. [8, 9] have developed formal operational and axiomatic models for subsets of the Intel-x86 and the Arm architectures. These models describe the persistency semantics

of the core instructions for interacting with non-volatile memory (namely, plain memory accesses and cache-line flush instructions). As can be seen in the cited publications, their semantics is often quite unexpected. Instructions execute asynchronously and even out-of-order, which leads to a number of possible weak persistency behaviors similar in natural to the weakly consistent behaviors observed by concurrent programs running on modern multicore processors. In the future, it would be useful to extend the existing models to cover additional features of the computer architectures, to extensively validate them with respect to hardware implementations, as well as to construct further characterizations of the persistency semantics of NVM programs using, say, denotational semantics with the overall goal at arriving at cleaner definitions.

Second, given that basic memory accesses have weak persistency semantics, how can programmers defend their programs against such weak behaviors, i.e., how can they *avoid weak persistency behaviors*? Currently, the only way to achieve this is to invoke special low-level hardware instructions, such as cache-line flush instructions (on x86 and Arm) or non-temporal store instructions (only on x86). Naively using these instructions, however, does not generate the desired result, because these instructions are weakly ordered with respect to other instructions. One therefore typically has to combine the usage of such instructions with (store) fence instructions to ensure their correct operation. Naturally, it would be desirable to define higher-level architecture-independent mechanisms for persisting stores to non-volatile memory, and to develop correct compilation schemes for these mechanisms to the various different platforms. Beyond higher-level fences and flushes, one can imagine developing even higher-level primitives, such as persistent transactions, persistent data structures.

Third, what are good *correctness criteria* for such persistent data structures and transactions? At a very basic level, one can require some basic algorithm-specific invariants to hold over the persistent state. For example, for a persistent sorted list, one may want to establish that the list is always well-formed in the persistent storage: its links point to valid fully initialized nodes, all such nodes are reachable from the head of the list, and the values of linked nodes are in ascending order. In addition, one may wish to establish an algorithm-independent correctness notion such as persistent serializability or linearizability, roughly stating that the implementation guarantees that the various operations executed in some total order, and that a prefix of that order has persisted.

Finally, given such appropriate correctness notions for persistent algorithms, what *techniques* can we used to establish correctness of such persistent algorithms? The work so far has considered mainly establishing invariants of the persisted state. Two approaches that have been tried are program logics and model checking. Specifically, Raad et al. [6] develop the Persistent Owicki-Gries (POG) program logic, an adaptation of the well-known Owicki-Gries proof system that is able to reason about invariants over the durable state under a subset of the Px86 semantics. In terms of model checking, Gorjiara et al. [4] have developed an approach for verifying assertions (such as invariants) of NVM programs. In a different context, Kokologiannakis et al. [5] developed a model checker for concurrent programs interacting with the ext4 file system, but their technique can easily be re-purposed to verify programs interacting with non-volatile memory. There has also been some works establishing persistent linearizability of simple durable algorithms (e.g., [2, 7]), but clearly much more work is needed to reach the level of complexity and automation that has been achieved for the verification of concurrent algorithms. Obvious next steps are: (1) to consider more advanced program logics, such as GPS [10] and FSL [3], which are extensions of separation logic that handle weak memory consistency, (2) to consider the complexity of basic verification questions following the initial results of Abdulla et al. [1], and (3) to develop automated techniques for checking and/or proving persistent linearizability.

In summary, NVM has created many research opportunities for our community. I intend to explore them in the near future and I hope that other researchers will join me in doing so – whether in collaboration or independently.

References

1. Parosh Aziz Abdulla, Mohamed Faouzi Atig, Ahmed Bouajjani, K. Narayan Kumar, and Prakash Saivasan. Deciding reachability under persistent x86-tso. *Proc. ACM Program. Lang.*, 5(POPL):1–32, 2021. doi:10.1145/3434337.
2. John Derrick, Simon Doherty, Brijesh Dongol, Gerhard Schellhorn, and Heike Wehrheim. Verifying correctness of persistent concurrent data structures: A sound and complete method. *Formal Aspects Comput.*, 33(4):547–573, 2021. doi:10.1007/s00165-021-00541-8.
3. Marko Doko and Viktor Vafeiadis. A program logic for C11 memory fences. In *VMCAI 2016*, volume 9583 of *LNCS*, pages 413–430. Springer, 2016. doi:10.1007/978-3-662-49122-5_20.
4. Hamed Gorjiara, Guoqing Harry Xu, and Brian Demsky. Jaaru: Efficiently model checking persistent memory programs. In *ASPLOS 2021*, pages 415–428. ACM, 2021. doi:10.1145/3445814.3446735.
5. Michalis Kokologiannakis, Ilya Kaysin, Azalea Raad, and Viktor Vafeiadis. PerSeVerE: Persistency semantics for verification under ext4. *Proc. ACM Program. Lang.*, 5(POPL):1–29, 2021. doi:10.1145/3434324.
6. Azalea Raad, Ori Lahav, and Viktor Vafeiadis. Persistent Owicki-Gries reasoning: A program logic for reasoning about persistent programs on Intel-x86. *Proc. ACM Program. Lang.*, 4(OOPSLA):151:1–151:28, 2020. doi:10.1145/3428219.
7. Azalea Raad and Viktor Vafeiadis. Persistence semantics for weak memory: Integrating epoch persistency with the TSO memory model. *Proc. ACM Program. Lang.*, 2(OOPSLA):137:1–137:27, 2018. doi:10.1145/3276507.
8. Azalea Raad, John Wickerson, Gil Neiger, and Viktor Vafeiadis. Persistency semantics of the intel-x86 architecture. *Proc. ACM Program. Lang.*, 4(POPL):11:1–11:31, 2020. doi:10.1145/3371079.
9. Azalea Raad, John Wickerson, and Viktor Vafeiadis. Weak persistency semantics from the ground up: Formalising the persistency semantics of ARMv8 and transactional models. *Proc. ACM Program. Lang.*, 3(OOPSLA):135:1–135:27, 2019. doi:10.1145/3360561.
10. Aaron Turon, Viktor Vafeiadis, and Derek Dreyer. GPS: Navigating weak memory with ghosts, protocols, and separation. In *OOPSLA 2014*, pages 691–707. ACM, 2014. doi:10.1145/2660193.2660243.

Initial Algebras Without Iteration

Jiří Adámek
Czech Technical University in Prague, Czech Republic
Technische Universität Braunschweig, Germany

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Lawrence S. Moss
Indiana University, Bloomington, IN, USA

Abstract

The Initial Algebra Theorem by Trnková et al. states, under mild assumptions, that an endofunctor has an initial algebra provided it has a pre-fixed point. The proof crucially depends on transfinitely iterating the functor and in fact shows that, equivalently, the (transfinite) initial-algebra chain stops. We give a constructive proof of the Initial Algebra Theorem that avoids transfinite iteration of the functor. For a given pre-fixed point A of the functor, it uses Pataraia's theorem to obtain the least fixed point of a monotone function on the partial order formed by all subobjects of A. Thanks to properties of recursive coalgebras, this least fixed point yields an initial algebra. We obtain new results on fixed points and initial algebras in categories enriched over directed-complete partial orders, again without iteration. Using transfinite iteration we equivalently obtain convergence of the initial-algebra chain as an equivalent condition, overall yielding a streamlined version of the original proof.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Logic and verification

Keywords and phrases Initial algebra, Pataraia's theorem, recursive coalgebra, initial-algebra chain

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.5

Category (Co)algebraic pearls

Funding *Jiří Adámek*: Supported by the grant No. 19-0092S of the Czech Grant Agency.
Stefan Milius: Supported by Deutsche Forschungsgemeinschaft (DFG) under project MI 717/7-1 and as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (393541319/GRK2475/1-2019).
Lawrence S. Moss: Supported by grant #586136 from the Simons Foundation.

1 Introduction

Owing to the importance of initial algebras in theoretical computer science, one naturally seeks results which give the existence of initial algebras in the widest of settings. We can distinguish two different, but related ideas which are commonly used in such results. By Lambek's Lemma, for every endofunctor $F\colon \mathscr{A} \to \mathscr{A}$, every initial algebra $\alpha\colon FA \to A$ has a structure α which is an isomorphism. So one might hope to obtain an initial algebra from a *fixed point*, viz. an F-algebra with isomorphic structure. It is sometimes much easier to find a *pre-fixed point*, an object A together with a monomorphism $m\colon FA \rightarrowtail A$. The Initial Algebra Theorem by Trnková et al. [28] states that, with inevitable but mild assumptions, any functor F which preserves monomorphisms and has a pre-fixed point also has an initial algebra. The proof uses the second prominent idea in the area: iteration, potentially into the transfinite. Indeed, transfinite iteration of F seems to be an essential feature of the proof.

The purpose of this paper is to prove the Initial Algebra Theorem in as wide a setting as possible with no use of iteration whatsoever. Moreover, the side conditions are mild: they apply, e.g. to the categories of complete metric spaces and directed-complete partial orders with a least element (shortly, dcpo with bottom). To situate our method in a larger context, recall that some fixed point theorems are proved with iteration, and some without. On the iterative side, we find Kleene's Theorem: continuous functions on ω-cpos with a least element have least fixed points obtained by iteration in countably many steps; and Zermelo's Theorem: monotone functions on chain-complete posets with a least element have least fixed points, using a transfinite iteration. On the non-iterative side, we have the Knaster-Tarski Theorem: monotone functions on complete lattices have both least and greatest fixed points, obtained by a direct definition without iteration. A relatively new result is Pataraia's Theorem: monotone functions on dcpos with bottom have a least fixed point. The latter two theorems are ordinal-free and indeed constructive.

The initial algebra for a functor F can often be constructed by iterating the functor, starting with the initial object 0 and obtaining a transfinite chain $0 \to F0 \to FF0 \cdots$ (Definition 6.2). The reason why fixed point theorems are useful for the proof of the Initial Algebra Theorem is that in every category $\mathscr{A}$, the collection $\mathsf{Sub}(A)$ of subobjects of a given object A is a partial order, and the iteration of F can be reflected by a particular monotone function $f\colon \mathsf{Sub}(A) \to \mathsf{Sub}(A)$ when $\alpha\colon FA \rightarrowtail A$ is a pre-fixed point; it takes a subobject $u\colon B \rightarrowtail A$ to $\alpha \cdot Fu$. If $\mathsf{Sub}(A)$ is sufficiently complete, then f has a least fixed point, and we show that this yields an initial algebra for F.

In order to make this step it is important for us that joins in $\mathsf{Sub}(A)$ are given by colimits in $\mathscr{A}$. Therefore Pataraia's Theorem is the best choice as a basis for the move from the least fixed point of f to the initial F-algebra. The reasons are that (a) it balances the weak assumption of monotonicity on f with the comparatively weak directed-completeness of the subobject lattice; and (b) its use yields an ordinal-free proof (in contrast to using Zermelo's Theorem, for which (a) is also the case). In fact, we present many examples of categories where directed joins of subobjects are given by colimits, while this is usually not the case for arbitrary joins, rendering the Knaster-Tarski Theorem a bad choice for us.

We start our exposition in Section 2.1 with a review of Pataraia's Theorem and also its non-constructive precursor, Zermelo's Theorem which we use later in Section 6. The second ingredient for the proof of our main result are recursive coalgebras, which we tersely review in Section 2.2. We use the fact that a recursive coalgebra which is a fixed point already is an initial algebra. Section 3 discusses the property that joins of subobjects are given by colimits. We make the technical notion of *smoothness* parametric in a class $\mathcal{M}$ of monomorphisms (representing subobjects), and we prove our results for a smooth class $\mathcal{M}$.

Our main result is the new proof of the Initial Algebra Theorem in Section 4. We apply it in Section 5 to the category $\mathsf{DCPO}_\perp$ of dcpos with bottom. The class of all embeddings is smooth. We derive a new result: if an endofunctor preserves embeddings and has a fixed point, then it has an initial algebra which coincides with the terminal coalgebra.

Finally, Section 6 rounds off our paper by providing the original Initial Algebra Theorem, which features the initial-algebra chain obtained by transfinite iteration. Although our proof has precisely the same mathematical content as the original one, it is slightly streamlined in that it appeals to Zermelo's Theorem rather than unfolding its proof.

Related work. Independently and at the same time, Pitts and Steenkamp [22] have obtained a result on the existence of initial algebras, which makes use of *sized functors* and is formalizable in Agda. In effect, they show that a form of iteration using sized functors is sufficient to obtain initial algebras. Our work, while constructive, is not aimed at formalization, and, as previously mentioned avoids iteration.

2 Preliminaries

We assume that readers are familiar with standard notions from the theory of algebras and coalgebras for an endofunctor F. We denote an initial algebra for F, provided it exists, by

$$\iota\colon F(\mu F) \to \mu F.$$

Recall that Lambek's Lemma [17] states that its structure ι is an isomorphism. This means that μF is a *fixed point* of F, viz. an object $A \cong FA$.

2.1 Fixed Point Theorems

In this subsection we present preliminaries on fixed point theorems for ordered structures. The most well-known such results are, of course, what is nowadays called Kleene's fixed point theorem and the Knaster-Tarski fixed point theorem. The former is for ω-cpos, partial orders with joins of ω-chains, with a least element (*bottom*, for short). Kleene's Theorem states that every endofunction which is *ω-continuous*, that is preserving joins of ω-chains, on an ω-cpo has a least fixed point. The Knaster-Tarski Theorem [16, 24], makes stronger assumptions on the poset but relaxes the condition on the endofunction. In its most general form it states that a monotone endofunction f on a complete lattice P has a least and greatest fixed point. Moreover, the fixed points of f form a complete lattice again.

Here we are interested in fixed point theorems that still work for arbitrary monotone functions but make do with weaker completeness assumptions on the poset P. One such result pertains to chain-complete posets. It should be attributed to Zermelo, since the mathematical content of the result appears in his 1904 paper [31] proving the Wellordering Theorem.

An *i-chain* in a poset P for an ordinal number i is a sequence $(x_j)_{j<i}$ of elements of P with $x_j \leq x_k$ for all $j \leq k < i$. The poset P is said to be *chain-complete* if every i-chain in it has a join. In particular, P has a least element $\bot$ (take $i = 0$).

Let $f\colon P \to P$ be a monotone map on the chain-complete poset P. Then we can define an ordinal-indexed sequence $f^i(\bot)$ by the following transfinite recursion:

$$f^0(\bot) = \bot, \quad f^{j+1}(\bot) = f(f^j(\bot)), \quad \text{and} \quad f^j(\bot) = \bigvee_{i<j} f^i(\bot) \quad \text{for limit ordinals } j. \tag{1}$$

It is easy to verify that this is a chain in P.

▶ **Theorem 2.1** (Zermelo). *Let P be a chain-complete poset. Every monotone map $f\colon P \to P$ has a least fixed point μf. Moreover, for some ordinal i we have $\mu f = f^i(\bot)$.*

Proof. Take i to be any ordinal larger than $|P|$, the cardinality of the set P. For this i, there must be some $j < i$ such that $f^j(\bot) = f^{j+1}(\bot)$. Indeed, this follows from Hartogs' Lemma [14], stating that for every set P there exists an ordinal i such that there is no injection $i \rightarrowtail P$. Thus, $f^j(\bot)$ is a fixed point of f. Let $f(x) = x$. An easy transfinite induction shows that $f^i(\bot) \leq x$ for all i. Hence, $f^j(\bot)$ is the least fixed point of f. ◀

There are also variations on Theorem 2.1, such as the result often called the Bourbaki-Witt Theorem [6, 30]; this states that every inflationary endo-map on a chain-complete poset has a fixed point above every element. (A map $f\colon P \to P$ is *inflationary*, if $x \leq f(x)$ for every $x \in P$.)

Theorem 2.1 is not constructive. Our proof relied on Hartogs' Lemma, which in turn builds on the standard theory of ordinals. That theory uses classical reasoning. A related point: some prominent results depending on ordinals are known to be unavailable in constructive set

theory (see [5]). For many of the end results, there is an alternative, Pataraia's Theorem [21], proved without iteration and without ordinals (see Theorem 2.4). This result is at the heart of this paper. It uses dcpos in lieu of chain-complete posets.

Pataraia sadly never published his result in written form. But it has appeared e.g. in work by Escardó [11], Goubault-Larrecq [13], Bauer and Lumsdane [5] based on a preprint by Dacar [9], and Taylor [27]. We present a proof based on Martin's presentation [19].

First recall that a *directed subset* of a poset P is a non-empty subset $D \subseteq P$ such that every finite subset of D has an upper bound in D. The poset P is called a *dcpo with bottom* if it has a least element and every directed subset $D \subseteq P$ has a join. Note that by Markowsky's Theorem [18], a poset is chain-complete iff it is a dcpo.

▶ **Remark 2.2.**

1. Observe that the set of all maps on a poset P form a poset using the point-wise order: $f \leq g$ if for every $x \in P$ we have $f(x) \leq g(x)$.
2. Hence, $f \colon P \to P$ is inflationary iff $\mathsf{id}_P \leq f$, where id_P is the identity function on P.
3. Function composition is *left-monotone*: we clearly have $f \cdot h \leq g \cdot h$ whenever $f \leq g$. Right-monotonicity additionally requires that the fixed argument be a monotone map: we have $f \cdot g \leq f \cdot h$ for every $g \leq h$ whenever f is monotone.
4. A monoid $(M, \cdot, 1)$ is *partially ordered* if M carries a partial order such that multiplication is monotone: $a \leq b$ and $a' \leq b'$ implies $a \cdot a' \leq b \cdot b'$. It is *directed complete* if it is a dcpo. An element $z \in M$ is a *zero* if $z \cdot m = z = m \cdot z$ for every $m \in M$.

▶ **Theorem 2.3** [19, Thm. 1]**.** *Every directed complete monoid $(M, \cdot, 1)$ whose bottom is the unit 1 has a top element which is a zero.*

Proof. The set M itself is directed: for $m, n \in M$ we see that $m \cdot n$ is an upper bound since

$$m = m \cdot 1 \leq m \cdot n \geq n \cdot 1 = n,$$

using that 1 is the bottom and multiplication is monotone. Thus M has a top element $\top = \bigvee M$. We have $\top = \top \cdot 1 \leq \top \cdot m$ for every $m \in M$, and clearly $\top \cdot m \leq \top$. Thus, $\top \cdot m = \top$ and, similarly $m \cdot \top = \top$, whence $\top$ is a zero. ◀

▶ **Theorem 2.4** (Pataraia's Theorem)**.** *Let P be a dcpo with bottom. Then every monotone map on P has a least fixed point.*

Proof.

1. Let M the set of all monotone inflationary maps on P. This is a monoid under function composition, with the unit id_P. Furthermore, M is a dcpo with bottom. Indeed, the order is the pointwise order from Remark 2.2, the least element is id_P, and directed joins are computed pointwise in P. Function composition is monotone (in both arguments) by Remark 2.2.3. By Theorem 2.3, M therefore has a top element $t \colon P \to P$ which is a zero.
2. Let $f \colon P \to P$ be inflationary and monotone. Then $f \in M$ and therefore $f \cdot t = t$. This means that for every $x \in P$, $f(t(x)) = t(x)$, whence $t(x)$ is a fixed point of f.
3. Now let $f \colon P \to P$ be just monotone. Let $\mathcal{S}$ be the collection of all subsets S of P which contain $\bot$, are closed under f, and under joins of directed subsets. (In more detail, we require that if $s \in S$, then $f(s) \in S$; and if $X \subseteq S$ is directed, $\bigvee X \in S$.) Clearly, $\mathcal{S}$ is closed under arbitrary intersections. Let $T = \bigcap \mathcal{S}$.
   The set of all post-fixed points $x \leq f(x)$ belongs to $\mathcal{S}$. Indeed, $\bot \leq f(\bot)$, and $f(x) \leq f(f(x))$ whenever $x \leq f(x)$. Moreover, a join $p = \bigvee D$ of a directed set D of post-fixed points of f is post-fixed point: p satisfies $d \leq f(d) \leq f(p)$ for every $d \in D$ due to the

monotonicity of f; thus, $p \leq f(p)$. By the minimality of T, we therefore know that T consists of post-fixed points of f. Thus, f restricts to a function $f\colon T \to T$. That restriction is inflationary (and monotone, of course) and therefore has a fixed point p by item 2.

4. We show that p is a least fixed point of $f\colon P \to P$. Suppose that x is any fixed point. The set $L = \{y \in P : y \leq x\}$ belongs to $\mathcal{S}$. Therefore $T \subseteq L$, which implies $p \leq x$. ◀

▶ **Corollary 2.5.** *The collection of all fixed points of a monotone map on a dcpo with bottom forms a sub-dcpo.*

This is analogous to fixed points of a monotone map on a complete lattice forming a complete lattice again, see Tarski [24].

Proof. Let f be monotone on the dcpo with bottom P. Put $S = \{x \in P : x = f(x)\}$. Suppose that $D \subseteq S$ be a directed subset, and let $w = \bigvee D$ be its join in P. Then we have $x = f(x) \leq f(w)$ for every $x \in S$ since f is monotone. Therefore $w \leq f(w)$ since w is the join of S. We now see that f restricts to $W = \{y \in P, w \leq y\}$, the set of all upper bounds of D in P: for every $y \in W$ we have $w \leq f(w) \leq f(y)$, which shows that $f(y) \in W$. Moreover, W is clearly a dcpo: it has least element w, and the join of every directed set of upper bounds of D is an upper bound, too. By Theorem 2.4, the restriction of f to W has a least fixed point p, say. In other words, p is the least fixed point of f among the upper bounds of D in P, and therefore it is the desired join of D in S. ◀

Here is our statement of a principle which we shall use later as a key step in our main result. It also appears in work by Escardó [11, Thm. 2.2] and Taylor [27].

▶ **Corollary 2.6** (Pataraia Induction Principle)**.** *Let P be a dcpo with bottom. If $f\colon P \to P$ is monotone, then μf belongs to every subset $S \subseteq P$ which contains $\bot$ and is closed under f and under directed joins.*

This follows from the proof of Theorem 2.4: items 3 and 4 show that $\mu f \in S$.

We apply the above principle to prove the following result that we will use in Section 5. A monotone function f on a dcpo D with bottom is *continuous* if it preserves directed joins, and *strict* if $f(\bot) = \bot$.

▶ **Lemma 2.7.** *Let P, Q be dcpos with bottom and let $f\colon P \to P$ and $g\colon Q \to Q$ be monotone. For every strict continuous map $h\colon P \to Q$ such that $g \cdot h = h \cdot f$ we have $h(\mu f) = \mu g$.*

Proof. First, $h(\mu f)$ is a fixed point of g: we have $g(h(\mu f)) = h(f(\mu f)) = h(\mu f)$. Therefore $\mu g \leq h(\mu f)$. For the reverse, let $S = \{x \in P : h(x) \leq \mu g\}$. Since h is strict, we see that $\bot \in S$. Moreover, S is closed under f, for if $x \in S$ we obtain $h(f(x)) = g(h(x)) \leq g(\mu g) = \mu g$ using monotonicity of g in the second step. Finally, S is closed under directed joins: if $D \subseteq S$ is a directed set we obtain $h(\bigvee D) = \bigvee_{x \in D} h(x) \leq \bigvee_{x \in D} \mu g = \mu g$, whence $\bigvee D$ lies in S. Thus, by Corollary 2.6, $\mu f \in S$, which means that $h(\mu f) \leq \mu g$. ◀

2.2 Recursive Coalgebras

A crucial ingredient for our new proof of the Initial Algebra Theorem are recursive coalgebras. They are closely connected to well-founded coalgebras and hence to the categorical formulation of well-founded induction. In his work on categorical set theory, Osius [20] first studied the notions of well-founded and recursive coalgebras (for the power-set functor on sets and, more

generally, the power-object functor on an elementary topos). He defined recursive coalgebras as those coalgebras $\alpha\colon A \to \mathscr{P}A$ which have a unique coalgebra-to-algebra homomorphism into every algebra (see Definition 2.8).

Taylor [25–27] considered recursive coalgebras for a general endofunctor under the name "coalgebras obeying the recursion scheme", and proved the General Recursion Theorem that all well-founded coalgebras are recursive for more general endofunctors; a new proof with fewer assumptions appears in recent work [2]. Recursive coalgebras were also investigated by Eppendahl [10], who called them algebra-initial coalgebras.

Capretta, Uustalu, and Vene [8] studied recursive coalgebras, and they showed how to construct new ones from given ones by using comonads. They also explained nicely how recursive coalgebras allow for the semantic treatment of recursive divide-and-conquer programs. Jeannin et al. [15] proved the general recursion theorem for polynomial functors on the category of many-sorted sets; they also provided many interesting examples of recursive coalgebras arising in programming.

In this section we will just recall the definition and a few basic results on recursive coalgebras which we will need for our proof of the initial algebra theorem.

▶ **Definition 2.8.** A coalgebra $\gamma\colon C \to FC$ is *recursive* if for every algebra $\alpha\colon FA \to A$ there exists a unique coalgebra-to-algebra morphism $h\colon C \to A$, i.e. a unique morphism h such that the square below commutes:

$$\begin{array}{ccc} C & \xrightarrow{h} & A \\ {\scriptstyle\gamma}\downarrow & & \uparrow{\scriptstyle\alpha} \\ FC & \xrightarrow{Fh} & FA \end{array} \tag{2}$$

Recursive coalgebras are regarded as a full subcategory of the category $\mathsf{Coalg}\,F$ of all coalgebras for the functor F.

▶ **Definition 2.9.** A *fixed-point* of an endofunctor is an object C together with an isomorphism $C \cong FC$. We consider C both as an algebra and a coalgebra for F.

▶ **Remark 2.10** [8, Prop. 7]**.** Every recursive fixed point is an initial algebra: for a coalgebra (C, γ) with γ invertible, the coalgebra-to-algebra morphisms from (C, γ) to an algebra (A, α) are the same as the algebra homomorphisms from (C, γ^{-1}) to (A, α).

▶ **Proposition 2.11** [8, Prop. 6]**.** *If (C, γ) is a recursive coalgebra, then so is $(FC, F\gamma)$.*

Proof. Let (A, α) be an algebra and denote by $h\colon C \to A$ the unique coalgebra-to-algebra morphism. We will show that

$$g = \big(FC \xrightarrow{Fh} FA \xrightarrow{\alpha} A\big)$$

is the unique coalgebra-to-algebra morphism from $(FC, F\gamma)$ to (A, α). First, diagram (2) for g commutes as can be seen on the left below:

$$\begin{array}{ccccc} & & \overset{g}{\frown} & & \\ FC & \xrightarrow{Fh} & FA & \xrightarrow{\alpha} & A \\ {\scriptstyle F\gamma}\downarrow & & \uparrow{\scriptstyle F\alpha} & & \uparrow{\scriptstyle\alpha} \\ FFC & \xrightarrow{FFh} & FFA & \xrightarrow{F\alpha} & FA \\ & & \underset{Fg}{\smile} & & \end{array} \qquad \begin{array}{ccccc} & & \overset{k\cdot\gamma}{\frown} & & \\ C & \xrightarrow{\gamma} & FC & \xrightarrow{k} & A \\ {\scriptstyle\gamma}\downarrow & & \downarrow{\scriptstyle F\gamma} & & \uparrow{\scriptstyle\alpha} \\ FC & \xrightarrow{F\gamma} & FFC & \xrightarrow{Fk} & FA \\ & & \underset{F(k\cdot\gamma)}{\smile} & & \end{array}$$

To see that g is unique, suppose that $k\colon FC \to A$ is a coalgebra-to-algebra morphism from $(FC, F\gamma)$ to (A, α). Then $k \cdot \gamma\colon C \to A$ is one from (C, γ) to (A, α). This is shown by the diagram on the right above. Thus, we have $h = k \cdot \gamma$, and we conclude that

$$g = \alpha \cdot Fh = \alpha \cdot Fk \cdot F\gamma = k,$$

where the last equation holds since k is a coalgebra-to-algebra morphism. ◀

▶ **Corollary 2.12.** *If a terminal recursive F-coalgebra exists, it is a fixed point of F.*

Indeed, the proof is the same as that for Lambek's Lemma, using Proposition 2.11 to see that for a terminal recursive coalgebra (T, τ), the coalgebra $(FT, F\tau)$ is recursive, too: the unique coalgebra homomorphism $h\colon (FT, F\tau) \to (T\tau)$ satisfies $h \cdot \tau = \mathsf{id}_T$ since $\tau\colon (T, \tau) \to (FT, F\tau)$ is a coalgebra homomorphism, and finally, $\tau \cdot h = Fh \cdot F\tau = F\,\mathsf{id}_T = \mathsf{id}_{FT}$.

▶ **Theorem 2.13** [8, Prop. 7]**.** *The terminal recursive coalgebra is precisely the same as the initial algebra.*

In more detail, let $F\colon \mathscr{A} \to \mathscr{A}$ be an endofunctor. Then we have:

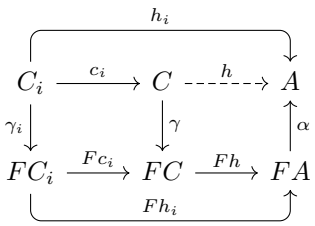
1. If (T, τ) is a terminal recursive coalgebra, then (T, τ^{-1}) is a initial algebra.
2. If $(\mu F, \iota)$ is an initial algebra, then $(\mu F, \iota^{-1})$ is a terminal recursive coalgebra.

Proof.

1. By Corollary 2.12, we know that τ is an isomorphism. By Remark 2.10, (T, τ^{-1}) is an initial algebra.
2. The coalgebra $(\mu F, \iota^{-1})$ is clearly recursive. It remains to verify its terminality. So let (C, γ) be a recursive coalgebra. There is a unique coalgebra-to-algebra morphism from (C, γ) to the algebra $(\mu F, \iota)$, and this means that there is a unique coalgebra homomorphism $h\colon (C, \gamma) \to (\mu F, \iota^{-1})$. ◀

▶ **Proposition 2.14.** *Every colimit of recursive coalgebras is recursive.*

Proof. We use the fact that the colimits in $\mathsf{Coalg}\,F$, the category of coalgebras for F, are formed on the level of the underlying category. Suppose that we are given a diagram of recursive coalgebras (C_i, γ_i), $i \in I$, with a colimit cocone $c_i\colon (C_i, \gamma_i) \to (C, \gamma)$ in $\mathsf{Coalg}\,F$. We prove that (C, γ) is recursive, too. Indeed, given an algebra (A, α) one takes for every i the unique coalgebra-to-algebra morphisms $h_i\colon (C_i, \gamma_i) \to (A, \alpha)$. Using unicity one sees that all h_i form a cocone of the diagram formed by all C_i in the underlying category. Therefore, there is a unique morphism $h\colon C \to A$ such that $h \cdot c_i = h_i$ holds for all $i \in I$. We now verify that h is the desired unique coalgebra-to-algebra morphism using the following diagram:



We know that the upper and lower parts, the left-hand square and the outside commute. Therefore so does the right-hand square when precomposed by every c_i. Since the colimit injections c_i form a jointly epic family, we thus see that the right-hand square commutes if and only if $h \cdot c_i = h_i$ holds for all $i \in I$. ◀

3 Smooth Monomorphisms

As we have just seen in Proposition 2.14, the collection of recursive coalgebras is closed under colimits. In order to apply an order-theoretic fixed point theorem to this collection, or to subcollections of it, we need a connection between colimits and subobjects. We make this connection by using the definition of *smooth class* of monomorphisms in a category.

For an object A of a category $\mathscr{A}$, a *subobject* is represented by a monomorphism $s\colon S \rightarrowtail A$. If s and $t\colon T \rightarrowtail A$ are monomorphisms, we write $s \leq t$ if s factorizes through t. If also $t \leq s$ holds, then t and s represent the same subobject; in particular S and T are then isomorphic. Generalizing a bit, let $\mathcal{M}$ be a class of monomorphisms. An *$\mathcal{M}$-subobject* of A is a subobject represented by a morphism $s\colon S \to A$ in $\mathcal{M}$. If the object A has only a set of subobjects, then we write

$$\mathsf{Sub}_{\mathcal{M}}(A)$$

for the poset of $\mathcal{M}$-subobjects of A.

If every object A only has a set of $\mathcal{M}$-subobjects, then $\mathscr{A}$ is called *$\mathcal{M}$-well-powered*.

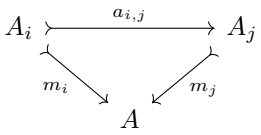
▶ **Definition 3.1.** Let $\mathcal{M}$ be a class of monomorphisms closed under isomorphisms and composition.

1. We say that an object A has *smooth $\mathcal{M}$-subobjects* provided that $\mathsf{Sub}_{\mathcal{M}}(A)$ is a dcpo with bottom (in particular, not a proper class) where the least element and directed joins are given by colimits of the corresponding diagrams of subobjects.
2. The class $\mathcal{M}$ is *smooth* if every object of $\mathscr{A}$ has smooth $\mathcal{M}$-subobjects.

Moreover, we say that a category has *smooth monomorphisms* if the class of all monomorphisms is smooth.

▶ **Remark 3.2.**

1. In more detail, let $D \subseteq \mathsf{Sub}_{\mathcal{M}}(A)$ be a directed set of subobjects represented by $m_i\colon A_i \rightarrowtail A$ $(i \in D)$. Then D has a join $m\colon C \rightarrowtail A$ in $\mathsf{Sub}_{\mathcal{M}}(A)$. Moreover, consider the diagram of objects $(A_i)_{i\in D}$ with connecting morphisms $a_{i,j}\colon A_i \rightarrowtail A_j$ for $i \leq j$ in D given by the unique factorizations witnessing $m_i \leq m_j$:



   (Note that $a_{i,j}$ need not lie in $\mathcal{M}$.) Then for every $i \in D$ there exists a monomorphism $c_i\colon A_i \rightarrowtail C$ with $m \cdot c_i = m_i$, since $m_i \leq m$. The smoothness requirement is that these monomorphisms form a colimit cocone.
2. Requiring that the least subobject in $\mathsf{Sub}_{\mathcal{M}}\, A$ is given by (the empty) colimit means that $\mathscr{A}$ has an initial object 0 and the unique morphism $0 \rightarrowtail A$ lies in $\mathcal{M}$.
3. If $\mathcal{M}$ is a smooth class, then $\mathscr{A}$ is $\mathcal{M}$-well-powered.

Since the above notion of smoothness is new, we discuss examples at length now. Below we show that in a number of categories the collection of all monomorphisms is smooth, as is the collection of all strong monomorphisms (those having the diagonal fill-in property with respect to epimorphisms). We also present some counterexamples and discuss other classes $\mathcal{M}$.

Recall the concept of a *locally finitely presentable* (*lfp*, for short) category (e.g. [4]): it is a cocomplete category $\mathscr{A}$ with a set of finitely presentable objects (i.e. their hom-functors preserve filtered colimits) whose closure under filtered colimits is all of $\mathscr{A}$. Examples are Set, Pos (posets and monotone maps), Gra (graphs and homomorphisms) and all varieties of finitary algebras such as monoids, vector spaces, rings, etc.

We say that $\mathscr{A}$ has a *simple* initial object 0 if all the morphisms with domain 0 are strong monomorphisms (equivalently, 0 has no proper quotients).

▶ **Example 3.3.** Both monomorphisms and strong monomorphisms are smooth in every lfp category with a simple initial object 0 [4, Cor. 1.63]. This includes Set, Pos, Gra, monoids and vector spaces. But not rings: in that category the initial object is $\mathbb{Z}$, the ring of integers, and there are non-monic ring homomorphisms with that domain (e.g. $\mathbb{Z} \to 1$).

▶ **Example 3.4.** Let us consider the category $\mathsf{DCPO}_\perp$ of dcpos with bottom and continuous maps between them, where a map is *continuous* if it is monotone and preserves directed joins.

1. In Section 5 we prove that the class of all embeddings (Definition 5.1) is smooth. (These play a major role in Smyth and Plotkin's solution method for recursive domain equations [23].) This example is one of several motivations for our move from the class of all monomorphisms to the more general situation of a class $\mathcal{M}$ in Definition 3.1.
2. In contrast, the class of all monomorphisms is non-smooth in $\mathsf{DCPO}_\perp$. For example, consider the dcpo $\mathbb{N}^\top$ of natural numbers with a top element $\top$. The subposets $C_n = \{0, \dots, n\} \cup \{\top\}$, $n \in \mathbb{N}$, form an ω-chain in $\mathsf{DCPO}_\perp$. Its colimit is $\mathbb{N}^\top \cup \{\infty\}$ where $n < \infty < \top$ for all $n \in \mathbb{N}$. The cocone of inclusion maps $C_n \hookrightarrow \mathbb{N}^\top$ consists of monomorphisms. However, the factorizing morphism from $\mathsf{colim}\, C_n$ to $\mathbb{N}^\top$ is not monic, as it merges ∞ and $\top$.
3. The same example demonstrates that strong monomorphisms are not smooth in $\mathsf{DCPO}_\perp$.

▶ **Example 3.5.**

1. Let us consider the category MS of metric spaces with distances at most 1 and *non-expanding* maps $f\colon (X, d_X) \to (Y, d_Y)$ (that is $d_Y(f(x), f(y)) \leq d_X(x, y)$ for all $x, y, \in X$. Although this category is not lfp, both monomorphisms and strong monomorphisms form smooth classes. The proof for strong monomorphisms is easy since the strong (equivalently, extremal) subobjects of a metric space A are represented by its subspaces (with the inherited metric). Given a directed set of subspaces $A_d \subseteq A$ ($d \in D$) their join in $\mathsf{Sub}(A)$ the subspace $\bigcup_{d \in D} A_d$ and this is also the colimit of the corresponding diagram in MS. The somewhat technical proof for monomorphisms is given in the appendix (Lemma A.1).
2. In the full subcategory CMS of MS given by all complete metric spaces monomorpisms are not smooth. This can be demonstrated as in Example 3.4.2: Let $\mathbb{N}^\top$ be the metric space with distances $d(n, m) = |1/2^{-n} - 1/2^{-m}|$ and $d(n, \top) = 1/2^{-n}$, and consider the ω-chain of spaces C_n where $d(n, \top) = 1$ and other distances are as in $\mathbb{N}^\top$.
3. In contrast, strong monomorphisms are smooth in CMS (see Lemma A.2).

The following equivalent formulation is often used in proofs.

▶ **Proposition 3.6.** *An object A has smooth $\mathcal{M}$-subobjects if and only if for every directed diagram D of monomorphisms in $\mathscr{A}$ (not necessarily members of $\mathcal{M}$), and every cocone $m_i\colon A_i \rightarrowtail A$, $i \in D$, of $\mathcal{M}$-monomorphisms, the following holds:*

1. *the diagram D has a colimit, and*
2. *the factorizing morphism induced by the cocone (m_i) is again an $\mathcal{M}$-monomorphism.*

Proof. The "only if" direction is obvious. For the "if" direction, suppose we are given a directed set $D \subseteq \mathsf{Sub}(A)$ of $\mathcal{M}$-subobjects $m_i\colon A_i \rightarrowtail A$ for $i \in D$ as in Remark 3.2. By item 1, the ensuing directed diagram of monomorphisms $a_{i,j}\colon A_i \rightarrowtail A_j$ has a colimit $c_i\colon A_i \to C$, $i \in D$, and we will prove that this yields the join $\bigvee_{i\in D} m_i$. By item 2, we have a unique $\mathcal{M}$-monomorphism $m\colon C \rightarrowtail A$ such that $m \cdot c_i = m_i$ for all $i \in D$.

Now let $s\colon S \rightarrowtail A$ be any $\mathcal{M}$-subobject with $m_i \leq s$ for all $i \in D$. That is, we have morphisms $s_i\colon A_i \to S$ with $s \cdot s_i = m_i$ for all $i \in D$. They form a cocone because for the monomorphism $a_{i,j}\colon A_i \rightarrowtail A_j$ witnessing $m_i \leq m_j$ we have

$$s \cdot s_j \cdot a_{i,j} = m_j \cdot a_{i,j} = m_i = s \cdot s_i,$$

whence $s_j \cdot a_{i,j} = s_i$ since s is monic. We therefore obtain a unique $t\colon C \to S$ with $t \cdot c_i = s_i$ for all $i \in D$. Consequently, we have

$$s \cdot t \cdot c_i = s \cdot s_i = m_i = m \cdot c_i \qquad \text{for all } i \in D.$$

Since the colimit injections c_i form an epic family, we conclude that $s \cdot t = m$, which means that $m \leq s$ in $\mathsf{Sub}_{\mathcal{M}}(A)$, as desired. ◀

▶ **Remark 3.7.**

1. Note that the conditions for $\mathcal{M}$ to be smooth are a part of the conditions of Taylor's notion of a *locally complete class of supports* [25, Def. 6.1. & 6.3] (see also [27, Assumption 4.18]).
2. Smoothness previously appeared for joins and colimit of chains in lieu of directed sets [2]. That formulation is related to the list of conditions for a class of monomorphisms given by Trnková et al. [28]. Note that a class $\mathcal{M}$ of monomorphisms containing the identities and closed under composition can be regarded as the subcategory of $\mathscr{A}$ given by all morphisms in $\mathcal{M}$. The list of conditions in op. cit. is equivalent to stating that the inclusion functor $\mathcal{M} \hookrightarrow \mathscr{A}$ creates colimits of chains. Requiring that the inclusion creates directed colimits implies that the class $\mathcal{M}$ is smooth. For the converse, we would need to add that for every directed diagram of $\mathcal{M}$-monomorphisms the colimit cocone consists of $\mathcal{M}$-monomorphisms.

4 The Initial Algebra Theorem

We are now ready to prove the main result of this paper.

▶ **Assumption 4.1.** Throughout this section we assume that $\mathscr{A}$ is a category with a class $\mathcal{M}$ of monomorphisms containing all isomorphisms and closed under composition. We say that $F\colon \mathscr{A} \to \mathscr{A}$ *preserves* $\mathcal{M}$ if $m \in \mathcal{M}$ implies $Fm \in \mathcal{M}$.

▶ **Definition 4.2.** An *$\mathcal{M}$-pre-fixed point* of F is an algebra whose structure $m\colon FA \rightarrowtail A$ lies in $\mathcal{M}$. In the case where $\mathcal{M}$ consists of all monomorphisms we speak of a pre-fixed point.

▶ **Theorem 4.3** (Initial Algebra Theorem)**.** *Let $m\colon FA \rightarrowtail A$ be an $\mathcal{M}$-pre-fixed point for an endofuctor preserving $\mathcal{M}$. If A has smooth $\mathcal{M}$-subobjects, then F has an initial algebra which is an $\mathcal{M}$-subalgebra of (A, m).*

Proof. We have the following endomap

$$f\colon \mathsf{Sub}_{\mathcal{M}}(A) \to \mathsf{Sub}_{\mathcal{M}}(A) \quad \text{defined by} \quad f\big(B \overset{u}{\rightarrowtail} A\big) = \big(FB \overset{Fu}{\rightarrowtail} FA \overset{m}{\rightarrowtail} A\big). \tag{3}$$

It is clearly monotone. We are going to apply Pataraia Induction to it. We take the subset $S \subseteq \mathsf{Sub}_{\mathcal{M}}(A)$ of all $u\colon B \rightarrowtail A$ such that $u \leq f(u)$ via some recursive coalgebra $\beta\colon B \to FB$. More precisely,

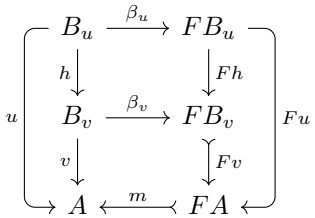
$$S = \{u\colon B \rightarrowtail A : u = m \cdot Fu \cdot \beta \text{ for some recursive coalgebra } \beta\colon B \rightarrowtail FB\}.$$

Note that if β exists for u, then it is unique. Moreover, $u \in S$ is a coalgebra-to-algebra morphism from (B, β) to (A, m).

The least subobject $0 \rightarrowtail A$ is clearly contained in S. Further, S is closed under f since $(FB, F\beta)$ is a recursive coalgebra by Proposition 2.11: for $u \in S$ we have

$$f(u) = m \cdot Fu = m \cdot F(m \cdot Fu \cdot \beta) = m \cdot F(f(u)) \cdot F\beta.$$

We continue with the verification that S is closed under directed joins. Let $D \subseteq S$ be directed. Given $u\colon B_u \rightarrowtail A$ in D we write $\beta_u\colon B_u \to FB_u$ for the recursive coalgebra witnessing $u \leq f(u)$. We show that these recursive coalgebras form a (then necessarily) directed diagram. To see this, we only need to prove that every morphism $h\colon B_u \rightarrowtail B_v$ witnessing $u \leq v$ in D; i.e. $v \cdot h = u$, is a coalgebra homomorphism. Consider the diagram below:



Since the outside, the lower square and the left-hand and right-hand parts commute, we see that the upper square commutes when extended by the monomorphism $m \cdot Fv$. Thus it commutes, proving that h is a coalgebra homomorphism.

Now denote by $v\colon B \rightarrowtail A$ the join $\bigvee D$ in $\mathsf{Sub}_{\mathcal{M}}(A)$. Since A has smooth subobjects, B is the colimit of the diagram formed by the B_u, $u \in D$, in $\mathscr{A}$. Since the forgetful functor $\mathsf{Coalg}\, F \to \mathscr{A}$ creates colimits, we have a unique coalgebra structure $\beta\colon B \to FB$ such that the colimit injections are coalgebra homomorphisms; moreover (B, β) is colimit of the coalgebras (B_u, β_u), $u \in D$. Thus, (B, β) is recursive by Proposition 2.14. Moreover, $v\colon B \rightarrowtail A$ is the unique morphism induced by the cocone given by all $u\colon B_u \rightarrowtail A$ in D. Since every $u \in S$ is the unique coalgebra-to-algebra morphism from (B_u, β_u) to (A, m), we know from the proof of Proposition 2.14 that v is the unique coalgebra-to-algebra morphism from (B, β) to (A, m). Thus, v lies in S.

By Theorem 2.4, f has a least fixed point, and by Corollary 2.6, $\mu f \in S$. Denote this subobject be $u\colon I \rightarrowtail A$. Since $u \in S$, there is a recursive coalgebra $\iota\colon I \to FI$ such that $u = m \cdot Fu \cdot \iota$. But u and $f(u) = m \cdot Fu$ represent the same subobject of A. So ι is an isomorphism. Thus (I, ι^{-1}) is an initial algebra by Remark 2.10. ◀

▶ **Corollary 4.4.** *Let $\mathscr{A}$ be a category with a smooth class $\mathcal{M}$ of monomorphisms. Then the following are equivalent for every endofunctor F preserving $\mathcal{M}$:*

1. *an initial algebra exists,*
2. *a fixed point exists,*
3. *an $\mathcal{M}$-pre-fixed point exists.*

Moreover, if these hold, then μF is an $\mathcal{M}$-subalgebra of every $\mathcal{M}$-pre-fixed point of F.

Indeed, Lambek's Lemma [17] tells us that 1 implies 2. Clearly, 2 implies 3 since $\mathcal{M}$ contains all isomorphisms. Theorem 4.3 shows that that 3 implies 1, and it also yields our last statement.

▶ **Corollary 4.5.** *Let $\mathscr{A}$ be an lfp category with a simple initial object. An endofunctor preserving monomorphisms has an initial algebra iff it has a pre-fixed point.*

▶ **Example 4.6.** We present examples which show, *inter alia*, that neither of the hypotheses in Theorem 4.3 can be left out. In each case $\mathcal{M}$ is the class of all monomorphisms.

1. The assumption that $0 \to A$ is monic. Let $\mathscr{A}$ be the variety of algebras (A, u, c) with unary operation u and a constant c. Its initial object is $(\mathbb{N}, s, 0)$ with $s(n) = n+1$, which is not simple. We present an endofunctor having no initial algebra even though it has a fixed point and preserves monomorphisms. Let $\mathscr{P}_0$ be the non-empty power-set functor. We obtain an analogous endofunctor $\bar{\mathscr{P}}_0$ on $\mathscr{A}$ defined by $\bar{\mathscr{P}}_0(A, u, c) = (\mathscr{P}_0 A, \mathscr{P}_0 u, \{c\})$. It clearly preserves monomorphisms, and the terminal object 1 is a fixed point of $\bar{\mathscr{P}}_0$ (since $\mathscr{P}_0 1 \cong 1$). This is, up to isomorphism, the only fixed point. However, it is not $\mu\bar{\mathscr{P}}_0$ because given an algebra on (A, u, c) with $u(x) \neq x$ for all $x \in A$, no $\bar{\mathscr{P}}_0$-algebra homomorphism exists from 1 to A.
2. The assumption that $\mathsf{Sub}(A)$ is a set in Definition 3.1.1. Let Ord be the totally ordered class of all ordinals taken as a category. In the opposite category $\mathsf{Ord}^{\mathsf{op}}$, all morphisms are monic, so every endofunctor preserves monomorphisms. For the functor F on $\mathsf{Ord}^{\mathsf{op}}$ given by $F(i) = i + 1$, every object is a pre-fixed point, and there are no fixed points. For each object i, $\mathsf{Sub}(i)$ has all the properties requested in Definition 3.1.1 except that it is a proper class.
3. Preservation of monomorphisms. Here we use the category $\mathsf{Set} \times \mathsf{Set}$ which satisfies all assumptions of Theorem 6.6. We define an endofunctor F by $F(X, Y) = (\emptyset, 1)$ if $X \neq \emptyset$ and $F(X, Y) = (\emptyset, \mathscr{P}Y)$ else. It is defined on morphisms as expected, using $\mathscr{P}$ in the case where $X = \emptyset$. This functor has many pre-fixed points, e.g. $F(1, 1) = (\emptyset, 1) \rightarrowtail (1, 1)$. But it has no fixed points (thus no initial algebra): first, $(\emptyset, Y)$ and $(\emptyset, \mathscr{P}Y)$ are never isomorphic, by Cantor's Theorem [7]. Second, if $X \neq \emptyset$, then there exists no morphism from (X, Y) to $F(X, Y) = (\emptyset, 1)$.

5 Initial Algebras in $\mathsf{DCPO}_\perp$-enriched Categories

It follows from the seminal paper by Smyth and Plotkin [23] that every locally continuous functor F on a category $\mathscr{A}$ enriched over ω-cpos (i.e. partial orders with a least element and joins of ω-chains) has an initial algebra $(\mu F, \iota)$ which is also a terminal coalgebra by inverting its structure. Local continuity means that the corresponding mappings $\mathscr{A}(A, B) \to \mathscr{A}(FA, FB)$ preserve (pointwise) directed joins. Here we assume the weaker property that F is locally monotone; for example, the endofunctor assigning to a dcpo its ideal completion is locally monotone, whence preserves embeddings, but not locally continuous. We apply Corollary 4.4 to derive that such an endofunctor has a pre-fixed point given by an embedding iff it has an initial algebra (being also the terminal coalgebra).

▶ **Definition 5.1.**

1. A category $\mathscr{A}$ is $\mathsf{DCPO}_\perp$-*enriched* provided that each hom-set is equipped with the structure of a dcpo with bottom, and composition preserves bottom and directed joins: for every morphism f and appropriate directed sets of morphisms g_i $(i \in D)$ we have

$$f \cdot \perp = \perp, \quad \perp \cdot f = \perp, \quad f \cdot \bigvee_{i \in D} g_i = \bigvee_{i \in D} f \cdot g_i, \quad \Big(\bigvee_{i \in D} g_i\Big) \cdot f = \bigvee_{i \in D} g_i \cdot f. \tag{4}$$

2. A functor on $\mathscr{A}$ is *locally monotone* if its restrictions $\mathscr{A}(A,B) \to \mathscr{A}(FA,FB)$ to the hom-sets are monotone.
3. A morphism $e\colon A \to B$ is called an *embedding* if there exists a morphism $\widehat{e}\colon B \to A$ such that $\widehat{e} \cdot e = \mathsf{id}_A$ and $e \cdot \widehat{e} \sqsubseteq \mathsf{id}_B$.

It is easy to see that the morphism $\widehat{e}$ is unique for e; it is called its *projection*.

The following result is a slight variation of a result by Smyth and Plotkin for ω-cpos [23]. We include the proof in the appendix for the convenience of the reader.

▶ **Theorem 5.2.** *Let D be a directed diagram of embeddings in a $\mathsf{DCPO}_\perp$-enriched category. For every cocone $(c_i\colon D_i \to C)$ of D, the following are equivalent:*

1. *The cocone (c_i) is a colimit.*
2. *Each c_i is an embedding, the composites $c_i \cdot \widehat{c}_i$ form a directed set in $\mathscr{A}(C,C)$, and*

$$\bigsqcup_i c_i \cdot \widehat{c}_i = \mathsf{id}_C\,. \tag{5}$$

▶ **Remark 5.3.** A $\mathsf{DCPO}_\perp$-enriched category $\mathscr{A}$ is $\mathcal{M}$-well-powered for the class $\mathcal{M}$ of all embeddings. The reason is that, given an object A, a subobject represented by an embedding $e\colon S \to A$ is determined by the endomorphism $e \cdot \widehat{e}$ on A. Indeed, let $f\colon T \to A$ be an embedding with $e \cdot \widehat{e} = f \cdot \widehat{f}$. Then $e = e \cdot \widehat{e} \cdot e = f \cdot \widehat{f} \cdot e$. Therefore, $e \leq f$ in $\mathsf{Sub}_{\mathcal{M}}(A)$. By symmetry $f \leq e$. Since $\mathscr{A}(A,A)$ is a set, $\mathcal{M}$-well-poweredness follows.

▶ **Theorem 5.4.** *Let $\mathscr{A}$ be a $\mathsf{DCPO}_\perp$-enriched category with directed colimits. Then the class of all embeddings is smooth.*

The proof is presented in Section A.3.

▶ **Corollary 5.5.** *Let $\mathscr{A}$ be a $\mathsf{DCPO}_\perp$-enriched category with directed colimits. For a locally monotone endofunctor F the following are equivalent:*

1. *an initial algebra exists,*
2. *a terminal coalgebra exists,*
3. *a fixed point exists.*

Moreover, if $(\mu F, \iota)$ is an initial algebra, then $(\mu F, \iota^{-1})$ is a terminal coalgebra.

Item 3 can be strengthened to state existence of a pre-fixed point carried by an embedding.

Proof. The dual category $\mathscr{A}^{\mathsf{op}}$ is $\mathsf{DCPO}_\perp$-enriched w.r.t. the same order on hom-sets. But the embeddings in $\mathscr{A}^{\mathsf{op}}$ are precisely the projections in $\mathscr{A}$. Every locally monotone endofunctor F on $\mathscr{A}$ clearly preserves embeddings and projections. Thus, the dual functor F^{op} on $\mathscr{A}^{\mathsf{op}}$ preserves embeddings. Now $1 \Leftrightarrow 3$ follows from an application of Corollary 4.4 to $\mathscr{A}$ and F, and $2 \Leftrightarrow 3$ is an application to $\mathscr{A}^{\mathsf{op}}$ and F^{op}. In each case the class $\mathcal{M}$ consists of all embeddings in $\mathscr{A}$ and $\mathscr{A}^{\mathsf{op}}$, respectively.

Finally, we prove that the initial algebra and terminal coalgebra coincide. Let $\iota\colon FI \to I$ be an initial algebra. Then we know that a terminal coalgebra $\tau\colon T \to FT$ exists. Moreover, from the last statement in Corollary 4.4 applied to F and its fixed point (T, τ^{-1}) we see that the unique F-algebra homomorphism $e\colon (I,\iota) \to (T,\tau^{-1})$ is an embedding. Another application of Corollary 4.4 to F^{op} and its fixed point (I,ι^{-1}) yields that the unique F^{op}-algebra homomorphism $f\colon \mu F^{\mathsf{op}} = (T,\tau) \to (I,\iota^{-1})$ is an embedding in $\mathscr{A}^{\mathsf{op}}$. This means that this an F-coalgebra homomorphism $f\colon (I,\iota^{-1}) \to (T,\tau)$ which is a projection in $\mathscr{A}$. By the universal properties of (I,ι) and (T,τ), $e = f$, and this morphism is both an embedding and a projections, whence an isomorphism. ◀

The requirement of local monotonicity of F can be weakened: the theorem holds for any endofunctor F which fulfils $Ff \sqsubseteq \mathsf{id}_{FA}$ whenever $f \sqsubseteq \mathsf{id}_A$. Indeed, a functor satisfying that property preserves embedding-projection pairs; in categories with split idempotents the converse holds, too [3, Obs. 6.6.5].

We close this section with a proposition on locally monotone functors which gives a version of a result for ω-cpo-enriched categories proved by Freyd [12] for locally continuous functors. He used Kleene's Theorem in lieu of Pataraia's.

▶ **Proposition 5.6.** *Let $\mathscr{A}$ be a $\mathsf{DCPO}_\perp$-enriched category. If a locally monotone functor F has an initial algebra $(\mu F, \iota)$, then $(\mu F, \iota^{-1})$ is a terminal coalgebra.*

We shall see in the proof that it is enough to assume that composition is left-strict: $\perp \cdot f = \perp$ holds for every morphism f of $\mathscr{A}$ (but $f \cdot \perp = \perp$ in (4) need not hold). This holds in categories typically used in semantics of programming languages, such as the category of dcpos with bottom and (non-strict) continuous maps, where composition is not (right-) strict.

Proof. Let $\iota\colon FI \to I$ be an initial algebra. For every coalgebra $\alpha\colon A \to FA$, we prove that a unique homomorphism into (I, ι^{-1}) exists.

1. Existence. The endomap f on $\mathscr{A}(A, I)$ given by $h \mapsto \iota \cdot Fh \cdot \alpha$ is monotone since F is locally monotone. Hence, it has a least fixed point $h\colon A \to I$ with $\iota^{-1} \cdot h = Fh \cdot \alpha$ by Pataraia's Theorem 2.4. This is a coalgebra homomorphism.
2. Uniqueness. First notice that for $\mathscr{A}(I, I)$ we have an the analogous endomap g given by $k \mapsto \iota \cdot Fk \cdot \iota^{-1}$. Since I is initial, the only fixed point of g is $k = \mathsf{id}_I$. Thus $\mathsf{id}_I = \mu g$. Now suppose that $h'\colon (A, \alpha) \to (I, \iota^{-1})$ is any coalgebra homomorphism. We know that $\mathscr{A}(h', I)\colon \mathscr{A}(I, I) \to \mathscr{A}(A, I)$ is a strict continuous map; strictness follows from left-strictness of composition: $\perp_{I,I} \cdot h' = \perp_{A,I}$. We now show that $g \cdot \mathscr{A}(h', I) = \mathscr{A}(h', I) \cdot f$. Indeed, unfolding the definitions, we have for every $k\colon I \to I$:

$$\begin{aligned} g \cdot \mathscr{A}(h', I)(k) &= g(k \cdot h') = \iota \cdot F(k \cdot h') \cdot \alpha = \iota \cdot Fk \cdot Fh' \cdot \alpha = \iota \cdot Fk \cdot \iota^{-1} \cdot h' \\ &= f(k) \cdot h' = \mathscr{A}(h', I)(f(k)). \end{aligned}$$

Therefore, by Lemma 2.7, $\mathscr{A}(h', I)(\mu f) = \mu g$, which means that $h' = \mathsf{id}_I \cdot h' = h$. ◀

We leave as an open problem to find an endofunctor on $\mathsf{DCPO}_\perp$ which has a fixed point but not an initial algebra.

6 The Initial-Algebra Chain

The proof of Theorem 4.3, relying on Pataraia's Theorem 2.4, is constructive. However, if one admits non-constructive reasoning and ordinals, then we can add another equivalent characterization to Corollary 4.4 in terms of the convergence of the initial-algebra chain, which we now recall.

▶ **Remark 6.1.**

1. Recall that an ordinal i is the (linearly ordered) set of all ordinals smaller than i. As such it is also a category.
2. By an *i-chain* in a category $\mathscr{C}$ is meant a functor $C\colon i \to \mathscr{C}$. It consists of objects C_j for all ordinals $j < i$ and (connecting) morphisms $c_{j,j'}\colon C_j \to C_{j'}$ for all pairs $j \leq j' < i$. Analogously, an *Ord-chain* in $\mathscr{C}$ is a functor from the totally ordered class Ord of all ordinals to $\mathscr{C}$. In both cases we will speak of a (transfinite) *chain* whenever confusion is unlikely.

3. A category $\mathscr{C}$ *has colimits of chains* if for every ordinal i a colimit of every i-chain exists in $\mathscr{C}$. (This does *not* include Ord-chains.) In particular, $\mathscr{C}$ has an initial object since the ordinal 0 is the empty set.

▶ **Definition 6.2** [1]**.** Let $\mathscr{A}$ be a category with colimits of chains. For an endofunctor F we define the *initial-algebra chain* $W\colon \mathsf{Ord} \to \mathscr{A}$. Its objects are denoted by W_i and its connecting morphisms by $w_{ij}\colon W_i \to W_j$, $i \leq j \in \mathsf{Ord}$. They are defined by transfinite recursion as follows

$W_0 = 0$, $\quad W_{j+1} = FW_j$ for all ordinals j, $\quad W_j = \mathsf{colim}_{i<j} W_i$ for all limit ordinals j,
$w_{0,1}\colon 0 \to W_1$ is unique, $\qquad w_{j+1,k+1} = Fw_{j,k}\colon FW_j \to FW_k$,
$w_{i,j}$ $(i < j)$ is the colimit cocone for limit ordinals j

▶ **Remark 6.3.**

1. There exists, up to natural isomorphism, precisely one Ord-chain satisfying the above equations. For example, $w_{\omega,\omega+1}\colon W_\omega \to FW_\omega$ is determined by the universal property of $W_\omega = \mathsf{colim}_{n<\omega} W_n = \mathsf{colim}_{n<\omega} W_{n+1}$ as the unique morphism with $w_{\omega,\omega+1} \cdot w_{n+1,\omega} = w_{n+1,\omega+1} = Fw_{n,\omega}$ for every $n < \omega$.
2. Every algebra $\alpha\colon FA \to A$ induces a canonical cocone $\alpha_i\colon W_i \to A$ $(i \in \mathsf{Ord})$ on the initial-algebra chain; it is the unique cocone with $\alpha_{i+1} = (W_{i+1} = FW_i \xrightarrow{F\alpha_i} FA \xrightarrow{\alpha} A)$ for all ordinals i. This is easy to see using transfinite induction.

▶ **Definition 6.4.** We say that the initial-algebra chain of a functor F *converges in λ steps* if $w_{\lambda,\lambda+1}$ is an isomorphism, and we simply say that it *converges*, if it converges in λ steps for some ordinal λ.

If $w_{i,i+1}$ is an isomorphism, then so is $w_{i,j}$, for all $j > \lambda$. This is easy to prove by transfinite induction.

Convergence of the initial-algebra chain yields an initial algebra [1]. We obtain this as a consequence of results from Section 2.2 on recursive coalgebras:

▶ **Theorem 6.5.** *Let $\mathscr{A}$ be a category with colimits of chains. If the initial-algebra chain of an endofunctor F converges in λ steps, then W_λ is the initial algebra with the algebra structure $w^{-1}_{\lambda,\lambda+1}\colon FW_\lambda \to W_\lambda$.*

Proof. An easy transfinite induction shows that every coalgebra $w_{i,i+1}\colon W_i \to FW_i$ is recursive: the coalgebra $0 \to F0$ is trivially recursive, for the isolated step use Proposition 2.11, and Proposition 2.14 yields the limit step. If $w_{\lambda,\lambda+1}$ is an isomorphism, then $(W_\lambda, w^{-1}_{\lambda,\lambda+1})$ is the initial algebra by Remark 2.10. ◀

The existence of an $\mathcal{M}$-pre-fixed point implies that the initial-algebra chain converges. The proof below is somewhat similar to the proof of Theorem 4.3. The difference is that one only uses the recursive coalgebras $W_i \to FW_i$ in the initial-algebra chain and applies Zermelo's Theorem 2.1 in lieu of Pataraia's Theorem. For this we work again under Assumption 4.1.

▶ **Theorem 6.6.** *Let F preserve $\mathcal{M}$ and $m\colon FA \rightarrowtail A$ be an $\mathcal{M}$-pre-fixed point. If A has smooth $\mathcal{M}$-subobjects, then the initial-algebra chain for F converges.*

Proof. Again, we use the monotone endomap $f\colon \mathsf{Sub}_{\mathcal{M}}(A) \to \mathsf{Sub}_{\mathcal{M}}(A)$ in (3). Theorem 2.1 applies since $\mathsf{Sub}_{\mathcal{M}}(A)$ is a dcpo by assumption, and therefore it is a chain-complete poset. Thus, f has the least fixed point $\mu f = f^i(\bot)$ for some ordinal i. The cocone $m_j\colon W_j \to A$ of Remark 6.3.2 satisfies $m_j = f^j(\bot)$ for all $j \in \mathsf{Ord}$. This is easily verified by transfinite induction. Hence, from $f(f^i(\bot)) = f^i(\bot)$ we conclude that m_i and m_{i+1} represent the same subobject of A. Since $m_i = m_{i+1} \cdot w_{i,i+1}$, it follows that $w_{i,i+1}$ is invertible, which means that the initial-algebra chain converges. ◀

We now obtain the original initial-algebra theorem by Trnková et al. [28]:

▶ **Corollary 6.7.** *Let $\mathscr{A}$ be a category with colimits of chains and with a smooth class $\mathcal{M}$ of monomorphisms. Then the following are equivalent for an endofunctor F preserving $\mathcal{M}$:*

1. *the initial-algebra chain converges,*
2. *an initial algebra exists,*
3. *a fixed point exists,*
4. *an $\mathcal{M}$-pre-fixed point exists.*

Moreover, if these hold, then μF is an $\mathcal{M}$-subalgebra of every $\mathcal{M}$-pre-fixed point of F.

Indeed, 4 implies 1 by Theorem 6.6, and 1 implies 2 is shown as in Theorem 6.5. The remaining implications are as for Corollary 4.4.

▶ **Remark 6.8.** Note that in lieu of assuming that $\mathscr{A}$ has colimits of *all* chains, it suffices that the initial-algebra chain exists (i.e. the colimits in Definition 6.2 exist). This weaker condition enables more applications, e.g. the category of relations with $\mathcal{M}$ the class of injective maps and functors F which are lifted from Set.

▶ **Remark 6.9.** For a set functor F no side condition is needed: if F has a pre-fixed point, then it has an initial algebra. This is clear if $F\emptyset = \emptyset$. If not, there is a set functor G with $G\emptyset \neq \emptyset$ which preserves monomorphisms and agrees with F on all nonempty sets and maps [29]. Since every pre-fixed point of F must be nonempty, it is also a pre-fixed point of G. Hence G has an initial algebra, which clearly is an initial algebra for F, too.

▶ **Corollary 6.10.** *An endofunctor on one of the categories* Set, Pfn, *or K-*Vec *has an initial algebra iff it has a pre-fixed point.*

Proof. For Set, use Remark 6.9. For Pfn and K-Vec, apply Corollary 6.7 with $\mathcal{M}$ the class of all monomorphisms (which are split and therefore preserved by every endofunctor). ◀

References

1 Jiří Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolin.*, 15:589–602, 1974.

2 Jiří Adámek, Stefan Milius, and Lawrence S. Moss. On well-founded and recursive coalgebras. In Barbara König and Jean Goubault-Larrecq, editors, *Proc. Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 12077 of *Lecture Notes Comput. Sci. (ARCoSS)*, pages 17–36. Springer, 2020.

3 Jiří Adámek, Stefan Milius, and Lawrence S. Moss. Initial algebras, terminal coalgebras, and the theory of fixed points of functors. draft book, available at `https://www8.cs.fau.de/ext/milius/publications/files/CoalgebraBook.pdf`, 2021.

4 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. Cambridge University Press, 1994.

5 Andrej Bauer and Peter Lefanu Lumsdane. On the Bourbaki-Witt principle in toposes. *Math. Structures Comput. Sci.*, 155(1):87–99, 2013.

6 Nicolas Bourbaki. Sur le théorème de Zorn. *Arch. Math.*, 2:434–437, 1949.

7 Georg Cantor. Über eine elementare frage der Mannigfaltigkeitslehre. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 1:75–78, 1891.

8 Venanzio Capretta, Tarmo Uustalu, and Varmo Vene. Recursive coalgebras from comonads. *Inform. and Comput.*, 204:437–468, 2006.

9 France Dacar. The join-induction principle for closure operators on dcpos. Available from `http://dis.ijs.si/France/`, January 2009.

10 Adam Eppendahl. Coalgebra-to-algebra morphisms. In *Proc. Category Theory and Computer Science (CTCS)*, volume 29 of *Electron. Notes Theor. Comput. Sci.*, pages 42–49, 1999.

11 Martín Escardó. Joins in the complete Heyting algebra of nuclei. *Appl. Categ. Structures*, 11:117–124, 2003.
12 Peter Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Applications of category theory in computer science: Proceedings of the London Mathematical Society Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1992.
13 Jean Goubault-Larrecq. Bourbaki, Witt, and Dito Pataraia. available at `https://projects.lsv.ens-cachan.fr/topology/?page_id=176`, June 2021.
14 Friedrich Hartogs. Über das Problem der Wohlordnung. *Math. Ann.*, 76(4):438–443, 1915.
15 Jean-Baptiste Jeannin, Dexter Kozen, and Alexandra Silva. Well-founded coalgebras, revisited. *Math. Structures Comput. Sci.*, 27:1111–1131, 2017.
16 Bronislav Knaster. Un theéorème sur les fonctions d'ensembles. *Ann. Soc. Pol. Math.*, 6:133–134, 1928.
17 Joachim Lambek. A fixpoint theorem for complete categories. *Math. Z.*, 103:151–161, 1968.
18 George Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6(1):53–68, 1976.
19 Keye Martin. Nothing can be fixed. In *Computation, logic, games, and quantum foundations*, volume 7860 of *Lecture Notes in Comput. Sci.*, pages 195–196. Springer, Heidelberg, 2013.
20 G. Osius. Categorical set theory: a characterization of the category of sets. *J. Pure Appl. Algebra*, 4(79–119), 1974.
21 Dito Pataraia. A constructive proof of Tarski's fixed-point theorem for dcpo's. Presented at the 65th Peripatetic Seminar on Sheaves and Logic, Aarhus, November 1997.
22 Andrew M. Pitts and S. C. Steenkamp. Constructing initial algebras using inflationary iteration, 2021. `arXiv:2105.03252`.
23 M. B. Smyth and G. D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM J. Comput.*, 11(4):761–783, 1982.
24 Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.
25 Paul Taylor. Towards a unified treatment of induction I: the general recursion theorem. preprint, available at `www.paultaylor.eu/ordinals/#towuti`, 1995–6.
26 Paul Taylor. *Practical Foundations of Mathematics*. Cambridge University Press, 1999.
27 Paul Taylor. Well founded coalgebras and recursion. available at `https://www.paultaylor.eu/ordinals/welfcr.pdf`, April 2021.
28 V. Trnková, J. Adámek, V. Koubek, and J. Reiterman. Free algebras, input processes and free monads. *Comment. Math. Univ. Carolin.*, 16:339–351, 1975.
29 Věra Trnková. On a descriptive classification of set functors I. *Comment. Math. Univ. Carolin.*, 12:143–174, 1971.
30 Ernst Witt. Beweisstudien zum Satz von M. Zorn. *Math. Nachr.*, 4:434–438, 1951.
31 Ernst Zermelo. Beweis, daß jede Menge wohlgeordnet werden kann. *Math. Ann.*, 59:514–516, 1904.

A Further Technical Details

A.1 Details for Example 3.5

▶ **Lemma A.1.** *Monomorphisms are smooth in* MS*.*

Proof. Fix a space (A, d), and consider a directed set D of subobjects $m_i\colon (A_i, d_i) \rightarrowtail (A, d)$ $(i \in D)$ with monomorphisms $a_{i,j}\colon A_i \rightarrowtail A_j$ witnessing $i \leq j$ in D. Let $B = \bigcup_{i\in D} m_i[A_i]$, and let $d'\colon B \to [0, 1]$ be defined as follows:

$$d'(x, y) = \inf\{d_i(x', y') : i \in D, x', y' \in A_i,\ m_i(x') = x \text{ and } m_i(y') = y\}. \tag{6}$$

We show that d' is a metric. It is clearly symmetric and fulfils $D'(x,x) = 0$. We verify that distinct points x, y in B have non-zero distance. For each i, x', and y' as in (6), $d_i(x', y') \geq d(x, y)$, since m_i is non-expanding. Thus $d'(x, y) \geq d(x, y) > 0$.

Finally, we verify that d' satisfies the triangle inequality. To this end it suffices to show that for all $x, y, z \in B$ and every $\varepsilon > 0$ we have $d'(x, z) \leq d'(x, y) + d'(y, z) + \varepsilon$.

Let $x, y, z \in B$ and fix $\varepsilon > 0$. We can choose i, x', y' as in (6) such that $d_i(x', y') < d'(x, y) + \varepsilon/2$. Analogously, let j, y'', z'' be such that $d_j(y'', z'') < d'(x, y) + \varepsilon/2$. Since the collection $m_i[A_i]$ is directed, we can assume $i \leq j$ in D. Using that the connecting map $a_{i,j}$ is non-expanding we obtain $d_j(a_{i,j}(x'), a_{i,j}(y')) < d'(x, y) + \varepsilon/2$. Since m_j is injective, $a_{i,j}(y') = y''$. Let $x'' = a_{i,j}(x')$, and note that $m_j(x'') = x$. By the triangle inequality in A_j,

$$d_j(x'', z'') \leq d_j(x'', y'') + d_j(y'', z'') < d'(x, y) + d'(y, z) + \varepsilon.$$

It follows that $d'(x, z) \leq d'(x, y) + d'(y, z) + \varepsilon$, as desired.

It is obvious that the inclusion $m\colon B \rightarrowtail A$ is non-expanding. It is also easy to check that (B, d') is the join in $\mathsf{Sub}(A, d)$ of the directed diagram corresponding to given directed set D.

Finally, for every $i \in D$, we have the codomain restriction $m_i'\colon A_i \rightarrowtail B$ of m_i, which is non-expanding. We verify that the family of all m_i' ($i \in D$) forms a colimit cocone. It clearly is a cocone. Consider any cocone $f_i\colon (A_i, d_i) \to (A^*, d^*)$, $i \in D$. Clearly, the union B is the colimit in Set. Therefore, we have a unique map $f\colon B \to A^*$ such that $f_i = f \cdot m_i'$ for all $i \in D$. This is given by $f(x) = f_i(x)$ whenever $x \in m_i[A_i]$. We check that f is non-expanding, and this will conclude our verification. Let $x, y \in B$, and choose i, x', y' as in (6). Since f_i is non-expanding,

$$d^*(f(x), f(y)) = d^*(f_i(x'), f_i(y')) \leq d_i(x', y') \leq d'(x, y). \qquad \blacktriangleleft$$

▶ **Lemma A.2.** *Strong monomorphisms are smooth in* CMS.

Proof. Fix a complete metric space (A, d), and consider a directed set D of closed subspaces $A_i \hookrightarrow A$. Their join $B \hookrightarrow A$ is the closure of their union

$$B = \overline{\bigcup_{i<\lambda} A_i}.$$

We know from Lemma A.1 that the union is the colimit of the directed diagram corresponding to D in MS. Moreover, the colimit of a diagram in CMS is given by forming the Cauchy completion of the colimit of that diagram in MS. (This follows from the fact that CMS is a reflective subcategory of MS with Cauchy completions as reflections.) Since B is complete and $\bigcup_{i\in D} A_i$ is dense in it, B is the Cauchy completion of that union, whence it is desired colimit in CMS. $\blacktriangleleft$

A.2 Proof of Theorem 5.2

Proof. $1 \Rightarrow 2$: Let D have objects D_i and connecting morphisms $e_{i,j}\colon D_i \to D_j$. Write $\widehat{e}_{i,j}$ for the projection of $e_{i,j}$. We verify that for $i \leq j \leq k$, $\widehat{e}_{i,k} = \widehat{e}_{i,j} \cdot e_{j,k}$. In fact, $\widehat{e}_{i,j}$ is unique with $\widehat{e}_{i,j} \cdot e_{i,j} = \mathsf{id}_{D_i}$ and $e_{i,j} \cdot \widehat{e}_{i,j} \sqsubseteq \mathsf{id}_{D_j}$. But $\widehat{e}_{i,k} \cdot e_{j,k}$ also has these properties, since

$$\begin{aligned}(\widehat{e}_{i,k} \cdot e_{j,k}) \cdot e_{i,j} &= \widehat{e}_{i,k} \cdot e_{i,k} \\ &= \mathsf{id}_{D_i},\end{aligned} \qquad \text{and} \qquad \begin{aligned}e_{i,j} \cdot (\widehat{e}_{i,k} \cdot e_{j,k}) &= e_{i,j} \cdot \widehat{e}_{i,j} \cdot \widehat{e}_{j,k} \cdot e_{j,k} \\ &= e_{i,j} \cdot \widehat{e}_{i,j} \cdot \mathsf{id}_{D_k} \\ &\sqsubseteq \mathsf{id}_{D_k}.\end{aligned}$$

This shows that indeed $\widehat{e}_{i,k} = \widehat{e}_{i,j} \cdot e_{j,k}$ for $i \leq j \leq k$.

For each i form the subdiagram D^i of all D_j for $j \geq i$, with connecting maps $e_{j,k}$ for $i \leq j \leq k$ inherited from D. Since I is directed, the colimit of D^i is $(c_j)_{j \geq i}$. Our observation at the outset shows that we have a cocone of D^i:

$$\widehat{e}_{i,j} = \big(D_j \xrightarrow{e_{j,k}} D_k \xrightarrow{\hat{e}_{i,k}} D_i\big).$$

Thus, there is a unique factorization $\widehat{c}_i \colon C \to D_i$ through the colimit cocone:

$$\widehat{e}_{i,j} = \widehat{c}_i \cdot c_j \quad \text{for } j \geq i. \tag{7}$$

In particular, for $i = j$, we see that $\widehat{c}_i \cdot c_i = \mathsf{id}_{D_i}$. We will verify below the equation $\bigsqcup_j c_j \cdot \widehat{c}_j = \mathsf{id}_C$, and this of course implies that $c_i \cdot \widehat{c}_i \sqsubseteq \mathsf{id}_C$. (This justifies our use of the projection notation $\widehat{c}_j$ and shows the first point in item 2, that c_i is an embedding.)

Next, we show that for each j, the morphisms $\widehat{c}_i$ for $i \leq j$ form a cone of D^j:

$$\widehat{c}_i = \big(C \xrightarrow{\hat{c}_j} D_j \xrightarrow{\hat{e}_{i,j}} D_i\big).$$

Indeed, the colimit cocone $(c_k)_{k \geq j}$ is collectively epic, so we need only establish this after precomposing with each c_k. We apply (7) twice to obtain: $\widehat{c}_i \cdot c_k = \widehat{e}_{i,k} = \widehat{e}_{i,j} \cdot \widehat{e}_{j,k} = \widehat{e}_{i,j} \cdot \widehat{c}_j \cdot c_k$.

We are ready to argue for 2. The maps $c_i \cdot \widehat{c}_i$ form a directed subset of $\mathscr{A}(C, C)$ because for $i \leq j$,$c_i \cdot \widehat{c}_i = (c_j \cdot e_{i,j}) \cdot (\widehat{e}_{i,j} \cdot \widehat{c}_j) \sqsubseteq c_j \cdot \widehat{c}_j$. Thus, $\bigsqcup_j c_j \cdot \widehat{c}_j$ exists. We use that the family (c_i) is collectively epic, and verify that $\bigsqcup_j c_j \cdot \widehat{c}_j \cdot c_i = c_i$ for every i. Fix i, and consider the join above. Since it is over a directed set, we need only consider $\bigsqcup_{j \geq i} c_j \cdot \widehat{c}_j \cdot c_i$. In addition, for $j \geq i$ we obtain $c_j \cdot \widehat{c}_j \cdot c_i = c_j \cdot \widehat{c}_j \cdot (c_j \cdot e_{i,j}) = c_j \cdot e_{i,j} = c_i$.

$2 \Rightarrow 1$: Let $(b_i \colon D_i \to B)$ be a cocone. For all $i \leq j$ we have $b_i = b_j \cdot e_{i,j}$. We also have a cocone (c_i), and so $c_i = c_j \cdot e_{i,j}$, and thus $\widehat{c}_i = \widehat{e}_{i,j} \cdot \widehat{c}_j$. From this we have

$$b_i \cdot \widehat{c}_i = (b_j \cdot e_{i,j}) \cdot (\widehat{e}_{i,j} \cdot \widehat{c}_j) \sqsubseteq b_j \cdot e_j.$$

Thus, the following join exists in $\mathscr{A}(C, B)$: $b = \bigsqcup_j b_j \cdot \widehat{c}_j$. To prove that $b_i = b \cdot c_i$ for all i, we fix one i and consider the join above with $j \geq i$:

$$b_j \cdot \widehat{c}_j \cdot c_i = (b_j \cdot \widehat{c}_j) \cdot (c_j \cdot e_{i,j}) = b_j \cdot e_{i,j} = b_i.$$

Thus $b \cdot c_i = \bigsqcup_{j \geq i} (b_j \cdot \widehat{c}_j \cdot c_i) = \bigsqcup_{j \geq i} b_i = b_i$. This shows that b is the desired factorization of (b_i). For its uniqueness, let $b' \colon C \to B$ be a morphism with $b' \cdot c_i = b_i$ for all i. Since $\bigsqcup_i c_i \cdot \widehat{c}_i = \mathsf{id}_C$, we have $b' = b' \cdot \big(\bigsqcup_i c_i \cdot \widehat{c}_i\big) = \bigsqcup_i b' \cdot c_i \cdot \widehat{c}_i = \bigsqcup_i b_i \cdot \widehat{c}_i = b$. This completes the proof. ◀

A.3 Proof of Theorem 5.4

Proof. We use Proposition 3.6. Fix an object A in $\mathscr{A}$. Let $\mathcal{E}$ be the class of embeddings, so that $\mathsf{Sub}_{\mathcal{E}}(A)$ denotes the poset of subobjects of A represented by embeddings. Let D be a directed diagram of monomorphisms in $\mathscr{A}$, not necessarily embeddings, and let $m_i \colon A_i \rightarrowtail A$, $i \in D$, be a cocone of morphisms in $\mathsf{Sub}_{\mathcal{E}}(A)$. By hypothesis, D has a colimit cocone, say $c_i \colon A_i \to B$. We have a unique morphism $m \colon B \to A$ such that for all i, $m_i = m \cdot c_i$. Our task is to show that m is an embedding, and that $m = \bigsqcup_{i \in A} m_i$ in $\mathsf{Sub}_{\mathcal{E}}(A)$.

For $i \leq j$ in A, we have a morphism $e_{i,j}$ such that $m_i = m_j \cdot e_{i,j}$. Let us verify that $e_{i,j}$ is an embedding and that $\widehat{e}_{i,j} = \widehat{m}_i \cdot m_j$. To see this, we use the characterization of projections.

First, $(\widehat{m}_i \cdot m_j) \cdot e_{i,j} = \widehat{m}_i \cdot m_i = \mathsf{id}$. Second, we verify $e_{i,j} \cdot (\widehat{m}_i \cdot m_j) \sqsubseteq \mathsf{id}$:

$$\begin{aligned} e_{i,j} \cdot (\widehat{m}_i \cdot m_j) &= (\widehat{m}_j \cdot m_j) \cdot e_{i,j} \cdot \widehat{m}_i \cdot m_j && \text{since } \widehat{m}_j \cdot m_j = \mathsf{id} \\ &= \widehat{m}_j \cdot m_i \cdot \widehat{m}_i \cdot m_j && \text{since } m_i = e_{i,j} \cdot m_j \\ &\sqsubseteq \widehat{m}_j \cdot m_j && \text{since } m_i \cdot \widehat{m}_i = \mathsf{id} \\ &= \mathsf{id}\,. \end{aligned}$$

We next show that for $i \leq j$, $c_i \cdot \widehat{m}_i \sqsubseteq c_j \cdot \widehat{m}_j$. Once this is done, we put $\widehat{m} = \bigsqcup_i c_i \cdot \widehat{m}_i$ and show that it is a projection for m. We thus calculate:

$$\begin{aligned} c_i \cdot \widehat{m}_i &= c_j \cdot e_{i,j} \cdot \widehat{m}_i \\ &= c_j \cdot e_{i,j} \cdot \widehat{m_j \cdot e_{i,j}} \\ &= c_j \cdot e_{i,j} \cdot \widehat{e}_{i,j} \cdot \widehat{m}_j \\ &\sqsubseteq c_j \cdot \widehat{m}_j \end{aligned}$$

To prove that $\widehat{m} \cdot m = \mathsf{id}$, we use that the family (c_i) is collectively epic. Thus, we show that for all i, $\widehat{m} \cdot m \cdot c_i = c_i$. We again consider $\bigsqcup_{j \geq i} c_j$ only:

$$\begin{aligned} \widehat{m} \cdot m \cdot c_i &= \big(\bigsqcup_{j \geq i} c_j \cdot \widehat{m}_j\big) \cdot m \cdot c_i \\ &= \bigsqcup_{j \geq i} c_j \cdot \widehat{m}_j \cdot m_i \\ &= \bigsqcup_{j \geq i} c_j \cdot \widehat{m}_j \cdot m_j \cdot e_{i,j} \\ &= \bigsqcup_{j \geq i} c_j \cdot e_{i,j} \\ &= \bigsqcup_{j \geq i} c_i \\ &= c_i. \end{aligned}$$

In the other direction, we show that $m \cdot \widehat{m} \sqsubseteq \mathsf{id}$:

$$m \cdot \big(\bigsqcup_i c_i \cdot \widehat{m}_i\big) = \bigsqcup_i m \cdot c_i \cdot \widehat{m}_i = \bigsqcup_i m_i \cdot \widehat{m}_i \sqsubseteq \mathsf{id}\,.$$

Our last order of business is to show that $m = \bigsqcup_{i \in A} m_i$ in $\mathsf{Sub}_{\mathcal{E}}(A)$. Since $m \cdot c_i = m_i$, we see that $m_i \sqsubseteq m$ for all i. Let $u \colon U \rightarrowtail A$ be an embedding with $m_i \sqsubseteq u$ for all i. Thus we have morphisms u_i such that $m_i = u \cdot u_i$. The family $(u_i)_i$ is a cocone of the original diagram D, because if $i \leq j$, then

$$u_j \cdot e_{i,j} = (\widehat{u} \cdot u) \cdot u_j \cdot e_{i,j} = \widehat{u} \cdot m_j \cdot e_{i,j} = \widehat{u} \cdot m_i = \widehat{u} \cdot u \cdot u_i = u_i.$$

Since (c_i) is a colimit, there is a unique $f \colon M \to U$ such that $u_i = f \cdot c_i$ for all i. We aim to show that $m = u \cdot f$, so that $m \sqsubseteq u$. For this, we again use the fact that (c_i) is a collectively epic family: $m \cdot c_i = m_i = u \cdot u_i = u \cdot f \cdot c_i$. ◀

Which Categories Are Varieties?

Jiří Adámek
Department of Mathematics, Czech Technical University in Prague, Czech Republic
Institute of Theoretical Computer Science, Technische Universität Braunschweig, Germany

Jiří Rosický
Department of Mathematics and Statistics, Faculty of Sciences,
Masaryk University, Brno, Czech Republic

Abstract

Categories equivalent to single-sorted varieties of finitary algebras were characterized in the famous dissertation of Lawvere. We present a new proof of a slightly sharpened version: those are precisely the categories with kernel pairs and reflexive coequalizers having an abstractly finite, effective strong generator. A completely analogous result is proved for varieties of many-sorted algebras provided that there are only finitely many sorts. In case of infinitely many sorts a slightly weaker result is presented: instead of being abstractly finite, the generator is required to consist of finitely presentable objects.

2012 ACM Subject Classification Theory of computation → Algebraic language theory

Keywords and phrases variety, many-sorted algebra, abstractly finite object, effective object, strong generator

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.6

Category (Co)algebraic pearls

Funding Supported by the grant No. 19-0092S of the Czech Grant Agency.

1 Introduction

In his dissertation in 1963 Lawvere not only introduced algebraic theories. He also characterized finitary single-sorted varieties as precisely those categories, up to equivalence, which have

(1) finite limits and coequalizers,
(2) effective equivalence relations, and
(3) an abstractly finite, regularly projective regular generator G.

Regular projectivity means that the hom-functor of G preserves regular epimorphisms. Abstract finiteness states that every morphism from G to a copower of G factorizes through a finite subcopower; this is much weaker than being finitely generated.

In Condition (1) of the dissertation coequalizers are not included. But they are used in the proof with no explanation, so this is just a small typo. Condition (2) can be avoided if a bit more than regular projectivity is required of G, as observed by Pedicchio und Wood [11]. We follow their idea and call an object G *effective* if its hom-functor preserves coequalizers of equivalence relations. Given a regularly projective regular generator G in a category $\mathcal{K}$, then it is effective iff $\mathcal{K}$ has effective equivalence relations (Proposition 25 below).

In software specification one typically works with many-sorted algebras, and the purpose of our paper is to generalize Lawvere's result to that case and improve it slightly: in (1) we need only to assume that kernel pairs and reflexive coequalizers exist. In (3) it is sufficient (in case of single-sorted varietie) to assume that G is an abstractly finite, effective, strong generator. For many-sorted varieties the concept of an *abstractly finite set* of objects was presented [1]. In case of finitely many sorts we obtain a completely analogous result to that above: a category with kernel pairs and reflexive coequalizers is equivalent to a finitary

S-sorted variety iff it has an abstractly finite strong generator formed by finitely many effective objects. The proof is based on the fact that for a monad T on $\mathbf{Set}^S$ to be finitary it is sufficient that it be *finitely bounded* (which, for S finite, has been shown in [3]). This means that every element of TX lies in the image of Tm for some finite S-sorted subset $m\colon M \hookrightarrow X$.

For infinitely many sorts an analogous, but slightly weaker result, is proved: in place of abstract finiteness one has to work with finitely presentable objects in the generator. In Example 39 below we demonstrate that the (expected) stronger result does not hold.

The results presented here are not substantially new. For example in [1] single-sorted varieties are characterized as cocomplete categories with a finitely generated, effective regular generator. But the message of our paper is that the beautiful result of Lawvere can be sharpened a little bit and extended to many-sorted varieties by applying the categorical methods developed in the subsequent 58 years. Many-sorted varieties have also been characterized as precisely the strongly locally presentable categories in [5]. We show that this result is an easy corollary of our main theorems.

Related Work. The concept of an abstractly finite set of objects was introduced in [1] and [2], where it was claimed that for many-sorted varieties the result of Lawvere completely generalizes. But the proof (based on the Birkhoff Variety Theorem) was not correct: see Example 39 which shows that the assumption of finitely many sorts is essential.

2 Abstractly Finite Objects

There are several concepts generalizing finite sets to "finite" objects of a category. Among the most important ones are that an object A of $\mathcal{K}$ is *finitely presentable* or *finitely generated* if its hom-functor $\mathcal{K}(A, -)$ preserves directed colimits (or directed colimits of monomorphisms, resp.). Lawvere [9] used in his characterization of varieties a weaker concept, abstract finiteness. He commented that it had been introduced by Peter Freyd.

We denote by

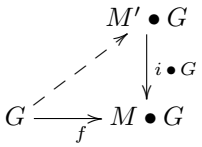
$$M \bullet G = \coprod_M G$$

the copower of an object G indexed by a set M. By a *subcopower* is meant the morphism

$$i \bullet M'\colon M' \bullet G \to M \bullet G$$

where $i\colon M' \hookrightarrow M$ is the inclusion map of a subset $M' \subseteq M$.

▶ **Definition 1.** *An object G is called abstractly finite if it has copowers, and every morphism $f\colon G \to M \bullet G$ factorizes through a finite subcopower:*



$(M' \subseteq M$ *finite*$)$

▶ Remark 2.

(1) Let G be an object with copowers. If it is finitely generated, then it is abstractly finite – but not vice versa, as we demonstrate in Example 3 below.

In fact, for every set $M \neq \emptyset$ we form a directed diagram of all finite non-empty subcopowers of $M \bullet G$. It has the colimit $M \bullet G$. Its connecting morphisms are

$j \bullet G\colon M' \bullet G \to M'' \bullet G$ for inclusion maps $j\colon M' \hookrightarrow M''$. Since j splits in **Set**, $j \bullet G$ is a split monomorphism. Thus the hom-functor of G preserves this directed colimit. Equivalently, G is abstractly finite.

(2) In a single-sorted variety of algebras, a free algebra G on a set X is abstractly finite iff X is finite. This follows easily from the fact that $M \bullet G$ is the free algebra on $M \times X$ for every set M.

▶ **Examples 3.**

(1) *In* **Set** *abstractly finite means finite. In the category of vector spaces it means finite-dimensional. So here finitely generated = abstractly finite.*

(2) *In the category of posets the abstractly finite objects are precisely the posets with finitely many connected components. Thus, an abstractly finite object can have an arbitrarily large cardinality.*
The same is true in the category of unary algebras on one operation or in the category of graphs.

(3) *In the category DCPO of dcpo's, i.e. posets with directed joins (and continuous maps) no nonempty object is finitely generated. In contrast, a dcpo is abstractly finite iff it has finitely many connected components.*

▶ Remark 4. Our focus is on varieties of S-sorted algebras, which we now shortly recall from [5, Chapter 14].

(1) By an S-sorted signature (for a set S) is meant a collection Σ of (operation) symbols σ with prescribed arities in $S^* \times S$. We write $\sigma\colon s_0 \dots s_{n-1} \to s$ if σ has arity $(s_1 \dots s_{n-1}, s)$.

(2) The category of S-sorted sets is denoted by $\mathbf{Set}^S$. Let $X \in \mathbf{Set}^S$ be an S-sorted set of a variables. The S-sorted set $F_\Sigma X$ of terms is the least one containing X and such that given an operation symbol $\sigma\colon s_0 \dots s_{n-1} \to s$ and terms p_i of sort s_i, then $\sigma(p_0, \dots, p_{n-1})$ is a (composite) term of sort s.

(3) A Σ-algebra is a sorted set A equipped with operations $\sigma_A\colon A_{s_0} \times \dots \times A_{s_{n-1}} \to A_s$ for every operation symbol $\sigma\colon s_0 \dots s_{n-1} \to s$. Given another Σ-algebra B, a homomorphism is a sorted map $f\colon A \to B$ preserving the operations: for every $\sigma\colon s_0 \dots s_{n-1} \to s$ we have

$$f_s \cdot \sigma_A = \sigma_B \cdot (f_{s_0} \times \dots \times f_{s_{n-1}})\,.$$

We denote by Σ- **Alg** the category of Σ-algebras and homomorphisms.

Example: $F_\Sigma X$ is a Σ-algebra w.r.t. composite terms as operations. This is a free Σ-algebra on X w.r.t. the inclusion map $\eta\colon X \hookrightarrow F_\Sigma X$.

(4) An *equation* using variables x_i of sort s_i $(i = 0, \dots, k-1)$ is an expression

$$\forall x_0 \dots \forall x_{k-1}(t = t')$$

where t, t' are terms in $F_\Sigma\{x_0, \dots, x_{n-1}\}$ of the same sort. A Σ-algebra A *satisfies* this equation if for every sorted function $f\colon \{x_0, \dots, x_{n-1}\} \to A$ the free homomorphism $\bar{f}\colon F_\Sigma\{x_i\} \to A$ fulfils $\bar{f}(t) = \bar{f}(t')$.

(5) For every set $\mathcal{E}$ of equations we denote by

$$(\Sigma, \mathcal{E})\text{- }\mathbf{Alg}$$

the full subcategory of Σ-**Alg** of all algebras satisfying all equations in $\mathcal{E}$. It is easy to see that this is a reflective subcategory of Σ-**Alg**, thus, it has free algebras on all sorted sets. And it is a complete and cocomplete category.

▶ **Definition 5.** *By a variety is meant a category* $(\Sigma, \mathcal{E})$-**Alg** *for some many sorted signature* Σ *and some set* $\mathcal{E}$ *of equations.*

▶ Remark 6. In a variety the usual meaning of "finitely generated", that is, having a finite set of generators, is equivalent to the categorical concept. The same is true about "finitely presentable", that is, presentable by finitely many generators and finitely many relations. See [4] 3.11 and 3.12.

For many-sorted varieties we need to generalize the concept of an abstractly finite object to sets of objects:

▶ **Definition 7.** *A set* $\mathcal{G}$ *of objects is abstractly finite if all coproducts of collections of objects from* $\mathcal{G}$ *exist, and given a morphism* $f\colon G \to \coprod_{i\in I} G_i$ *with* G *and all* G_i *in* $\mathcal{G}$*, then* f *factorizes through a finite subcoproduct of* $\coprod_{i\in I} G_i$.

The factorization of f is not required to be unique (and coproduct injections are not required to be monic).

▶ **Example 8.** In an S-sorted variety we form the free algebra G_s on one generator of sort s for each $s \in S$. The set $\{G_s\}_{s\in S}$ is abstractly finite. Indeed, the coproduct $\coprod_{i\in I} G_{s_i}$ is precisely a free algebra on the S-sorted set X with $X_s = \{i \in I, s_i = s\}$. Every homomorphism $f\colon G_s \to \coprod_{i\in I} G_{s_i}$ maps the generator $x \in G_s$ to a term over X. If $Y \subseteq X$ is the set of all variables that appear in the term $f(x)$, then $f(x)$ lies in the finite subcoproduct $\coprod_{s_i\in Y} G_{s_i}$. Consequently, f factorizes through this subcoproduct.

▶ Remark 9. Recall that a *strong generator* is a set $\mathcal{G}$ of objects such that coproducts of collections of objects from $\mathcal{G}$ exist, and for every object X there exists an epimorphism $c\colon \coprod_{i\in I} G_i \to X$ with $G_i \in \mathcal{G}$ for $i \in I$ which is extremal (i.e., c does not factorize through any proper subobject of X).

A *regular generator* has the stronger property that the following canonical morphism

$$c_X = [f]\colon \coprod_{G\in\mathcal{G}} \coprod_{f\colon G\to X} G \to X$$

is a regular epimorphism.

▶ **Example 10.**

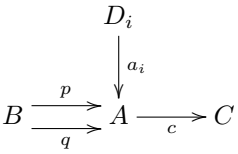
(1) In a single-sorted variety $\mathcal{K}$ the free algebra G on one generator is an abstractly finite regular generator. Indeed, for every algebra X the coproduct $\coprod_{f\colon G\to X} G$ is the free algebra of $\mathcal{K}$ generated by $\mathcal{K}(G, X)$, and the canonical morphism c_X is surjective, i.e., a regular epimorphism.

(2) In an S-sorted variety the set $\{G_s\}_{s\in S}$ from Example 8 is an abstractly finite regular generator. The argument is as above.

▶ Remark 11.

(1) In the next theorem we use the standard construction of colimits via coproducts and coequalizers [10] Thm. V.2.2. Let $D\colon \mathcal{D} \to \mathcal{K}$ be a diagram with objects D_i $(i \in I)$. Suppose that the following coproducts exist: $A = \coprod_{i\in I} D_i$ with injections a_i, and $B = \coprod_{f\colon D_i\to D_j} D_i$ with injection b_f, where the f's range over all morphisms of $\mathcal{D}$. Then

we form the morphisms p, $q\colon B \to A$ with f-components a_i and $a_j \cdot f$, resp. for all $f\colon D_i \to D_j$. Suppose a coequalizer c of p and q exists:



Then the cocone of $c \cdot a_i$ $(i \in I)$ is a colimit of D.

(2) We recall the concept of dense subcategory. Given a full subcategory $\mathcal{G}$ of $\mathcal{K}$, for every object K we form the slice category $\mathcal{G}/K$ of all morphisms $f\colon G \to K$ with $G \in \mathcal{G}$. The forgetful functor $D_K\colon \mathcal{G}/K \to \mathcal{K}$ sending $f\colon G \to K$ to G has the cocone formed by all f's. Then $\mathcal{G}$ is *dense* if this cocone is a colimit of D_K (for every object K).

The proof of the following theorem uses ideas of Lawvere's thesis [9]. A shorter proof presented in [1] was not correct.

▶ **Theorem 12.** *Let G be an abstractly finite, regular, singleton generator. Then the full subcategory of finite copowers of G is dense.*

Proof. Denote by $\mathcal{G}$ the full subcategory of all $n \bullet G$, $n \in \mathbb{N}$. For every object K we prove that $K = \mathit{colim}\, D_K$. In detail, given an object L and a cocone of D_K denoted by $(-)'$:

$$\frac{n \bullet G \xrightarrow{f} K}{n \bullet G \xrightarrow[f']{} L} \qquad (n \in \mathbb{N})$$

we prove that there exists a unique morphism $h\colon K \to L$ with $f' = h \cdot f$ for all f. The fact that $(-)'$ is a cocone means that

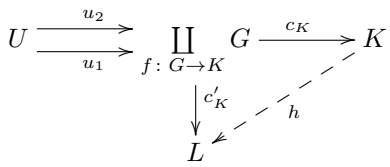
$$f = g \cdot u \quad \text{implies} \quad f' = g' \cdot u \tag{2.1}$$

for all morphisms $u\colon n \bullet G \to m \bullet G$ and $g\colon m \bullet G \to K$. In particular, if $f_i\colon G \to K$ are the components of $f\colon n \bullet G \to K$ $(i = 1, \ldots, n)$, then we get the corresponding morphisms $f_i'\colon G \to L$. We then obtain

$$f' = \left[f_1', \ldots, f_n'\right] \tag{2.2}$$

by applying (2.1) to the coproduct injections on $n \bullet G$.

The canonical morphism $c_K\colon \coprod_{f\colon G \to K} G \to K$ is a coequalizer of a pair u_1, $u_2\colon U \to \coprod_{f\colon G \to K} G$. Denote by $c_K'\colon \coprod_{f\colon G \to K} G \to L$ the morphism with components f' for every $f\colon G \to K$:

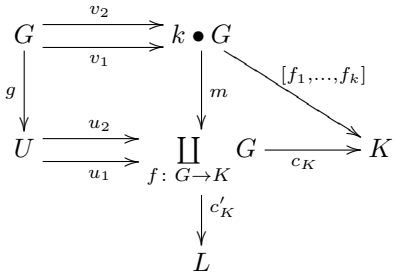


We are going to prove that $c_K' \cdot u_1 = c_K' \cdot u_2$. Then we obtain a factorization h with $c_K' = h \cdot c_K$. This is the desired morphism: for every $f\colon G \to K$ we then have $f = h \cdot f'$, and due to (2.2)

the same holds for every $f\colon n \bullet G \to K$. Uniqueness of h is clear since G is a generator. For proving $c'_K \cdot u_1 = c'_K \cdot u_2$ we just need to verify

$$c'_K \cdot u_1 \cdot g = c'_K \cdot u_2 \cdot g \quad \text{for every} \quad g\colon G \to U\,. \tag{2.3}$$

The morphisms $u_1 g$, $u_2 g\colon G \to \coprod_{f\colon G \to K} G$ both factorize through a finite subcopower, since G is abstractly finite. That is, we have $f_i\colon G \to K$ for $i = 1, \ldots, k$ such that for the corresponding coproduct injection $m\colon k \bullet G \to \coprod_{f\colon G \to K} G$ there exist morphisms v_i with $g \cdot u_i = m \cdot v_i$:



From $c_K \cdot u_1 = c_K \cdot u_2$ we get $c_K \cdot m \cdot v_1 = c_K \cdot m \cdot v_2$, thus, $(c_K \cdot m \cdot v_1)' = (c_K \cdot m \cdot v_2)'$. By applying (2.1) to the morphisms mv_i of $\mathcal{G}$ we get

$$c'_K u_i g = c'_K m v_i = (c_K m v_i)' = (c_K u_i g)'$$

for $i = 1, 2$, which proves (2.3). ◀

▶ Remark 13. The above theorem and proof immediately generalize to non-singleton abstractly finite regular generators $\mathcal{G}$: the closure of $\mathcal{G}$ under finite coproducts is dense.

▶ Remark 14. A pair of morphisms f_1, $f_2\colon A \to B$ is called *reflexive* if there exists $d\colon B \to A$ with $f_1 \cdot d = f_2 \cdot d = \mathrm{id}_B$. A category is said to have *reflexive coequalizers* if every reflexive pair has a coequalizer.

▶ **Corollary 15.** *Let $\mathcal{K}$ be a category with reflexive coequalizers. Then it is complete and cocomplete provided that it has an abstractly finite regular generator consisting of regular projectives.*

This follows from Remark 13. Let $\bar{\mathcal{G}}$ be the dense closure of $\mathcal{G}$. Consequently, the functor $E\colon \mathcal{K} \to \left[\bar{\mathcal{G}}^{\mathrm{op}}, \mathbf{Set}\right]$ assigning to K the restriction of $\mathcal{K}(-, K)$ to $\bar{\mathcal{G}}^{\mathrm{op}}$ is full and faithful. Moreover E has a left adjoint: it assigns to $H\colon \bar{\mathcal{G}}^{\mathrm{op}} \to \mathbf{Set}$ the colimit of the category of elements of H. The reason why this colimit exists is that in Remark 11(1) the two coproducts exist, since they are formed by object of $\bar{\mathcal{G}}^{\mathrm{op}}$ (and so they are coproducts of objects of $\mathcal{G}$), and the coequalizer c exists because the pair p, $q\colon B \to A$ is reflexive. Indeed, the morphism $d\colon A \to B$ whose i-component is the coproduct injection corresponding to id_{D_i} fulfils $p \cdot d = q \cdot d = \mathrm{id}_A$. The details why this yields a left adjoint of E can be found in [1, Corollary 2.12].

We conclude that $\mathcal{K}$ is equivalent to a full reflective subcategory of $[\bar{\mathcal{G}}^{\mathrm{op}}, \mathbf{Set}]$, hence, it is complete and cocomplete.

3 Effective Objects

A category is said to have effective equivalence relations if every equivalence relation (see below) is the kernel pair of its coequalizer. We define effective objects as those whose hom-functors preserve coequalizers of equivalence relations. Based on an idea of Pedicchio and Wood [11] we then prove that given a regularly projective regular generator G, it is effective iff equivalence relations are effective.

The usual definition of a relation R on an object A is: a subobject of $A \times A$. The restricted projections then form a parallel pair $r_1, r_2 \colon R \to A$ of morphisms which is collectively monic. Our main theorem makes no assumptions about the existence of products. We therefore introduce relations via parallel pairs:

▶ **Definition 16** ([7] 2.5.2). *Let A be an object of a category $\mathcal{K}$.*

(1) *A relation on A is represented by a collectively monic ordered pair of morphisms r_1, $r_2 \colon R \to A$. Another such pair r_1', $r_2' \colon R' \to A$ represents the same relation iff there is an isomorphism $i \colon R' \to R$ with $r_1' = r_1 i$ and $r_2' = r_2 i$.*
We speak about "the relation R" if r_1, r_2 are clear.

(2) *A relation R is an equivalence if for every object X of $\mathcal{K}$ the following relation on the hom-set $\mathcal{K}(X, A)$ is an equivalence relation in the usual sense:*

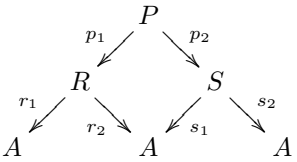
$$\{(r_1 f, r_2 f); f \colon X \to R\}.$$

▶ Remark 17. Let $\mathcal{K}$ have finite limits and *regular factorizations* (every morphism factorizes as a regular epimorphism followed by a monomorphism). Then equivalence relations are more intuitive:

(1) A relation R on A is precisely a subobject of $A \times A$.
Example: Δ_A given by id_A, id_A.

(2) Every parallel pair of morphisms f_1, $f_2 \colon B \to A$ represents a relation on A : factorize $\langle f_1, f_2 \rangle \colon B \to A \times A$ as a regular epimorphism $e \colon B \to R$ followed by a monomorphism $\langle r_1, r_2 \rangle \colon R \to A \times A$. This gives you a relation $R \rightarrowtail A \times A$.

(3) A *composite* of relations r_1, $r_2 \colon R \to A$ and s_1, $s_2 \colon S \to A$ is the relation $P = R \circ S$ represented by the pair $(r_1 p_1, s_2 p_2)$ for the following pullback P of r_2 and s_1 :



(4) The relation R^0 is represented by r_2, $r_1 \colon R \to A$.

(5) A relation R on A is an equivalence relation iff it is reflexive ($\Delta_A \subseteq R$), symmetric ($R^0 \subseteq R$) and transitive ($R \circ R \subseteq R$). This is equivalent to the definition above by [7] Proposition 2.5.1.

(6) For every morphism $f \colon A \to B$ the *kernel pair*, which means the pullback of two copies of f, is an equivalence relation. By an *effective equivalence* is meant the kernel pair of some morphism.

(7) A functor $E \colon \mathcal{K} \to \mathcal{L}$ is said to *reflect isomorphisms* if whenever Eh is invertible in $\mathcal{L}$, then h is invertible in $\mathcal{K}$. Analogously for reflecting regular epimorphisms, limits, etc.

Suppose that E preserves and reflects (a) finite limits and (b) regular factorizations. Then it preserves and reflects relations and relation composition. Since the operation $R \mapsto R^0$ is also preserved and reflected, we conclude that E preserves and reflects equivalence relations.

▶ **Definition 18.** *A category is said to have effective equivalence relations if every equivalence on an object A is effective (the kernel equivalence of some morphism $f\colon A \to B$).*

▶ **Example 19.** Let $\mathcal{K}$ be a variety. A relation on an algebra A is a subalgebra of $A \times A$. And an equivalence is precisely a congruence R on A, represented by its projections r_1, $r_2\colon R \to A$. Every variety has effective equivalence relations.

▶ **Definition 20.** *An object A is called effective if its hom-functor $\mathcal{K}(A, -)$ preserves coequalizers of equivalence relations.*

▶ **Example 21.** In a variety $\mathcal{K}$ all free algebras are effective. In fact, let us first consider a single-sorted variety. Its forgetful functor $U\colon \mathcal{K} \to \mathbf{Set}$ preserves coequalizers of congruences. (Indeed, if R is a congruence on A, then the quotient algebra A/R is formed on the corresponding quotient set. And the canonical map $c\colon A \to A/R$ is precisely the coequalizer of the projections r_1, $r_2\colon R \to A$.) Now $U \simeq \mathcal{K}(G, -)$ where G is the free algebra on one generator. Thus, G is effective. And since a free algebra on a set M is precisely $M \bullet G$, its hom-functor is the M-copower of U, and it also preserves coequalizers of congruences.

The argument for S-sorted varieties is analogous, using that the forgetful functor $U\colon \mathcal{K} \to \mathbf{Set}^S$ preserves coequalizers of congruences.

▶ **Definition 22.** *An object G is called regularly projective if its hom-functor preserves regular epimorphisms. More detailed: given a regular epimorphism $e\colon A \to B$, every morphism from G to B factorizes through e.*

▶ Remark 23. Let $\mathcal{K}$ be a category with kernel pairs. Recall that kernel pairs are called effective equivalences.

(1) Every effective object G is regularly projective.

In fact, given a regular epimorphism $e\colon A \to B$ form its kernel pair r_1, $r_2\colon R \to A$. Then $\mathcal{K}(G, e)$ is a coequalizer of $\mathcal{K}(G, r_i)$, thus it is surjective.

(2) An object G is regularly projective iff its hom-functor preserves coequalizers of effective equivalences – thus, this is very "near" to being effective.

Indeed, let r_1, $r_2\colon R \to A$ be an effective equivalence. Then the coequalizer $e\colon A \to B$ has the kernel equivalence R. Since $\mathcal{K}(G, -)$ preserves pullbacks, $\mathcal{K}(G, e)$ has the kernel pair $\mathcal{K}(G, r_i)$, $i = 1, 2$ in $\mathbf{Set}$. We know that $\mathcal{K}(G, e)$ is surjective. This implies that this is the coequalizer of its kernel pair.

▶ Remark 24. Let $\mathcal{K}$ be a category with kernel pairs and their coequalizers and let $\mathcal{G}$ be a regular generator formed by regular projectives.

(1) $\mathcal{K}$ has regular factorizations: every morphism f factorizes as a regular epimorphism followed by a monomorphism. Indeed, let p_1, $p_2\colon P \to A$ be the kernel pair of $f\colon A \to B$ and $e\colon A \to C$ be a coequalizer of p_1, p_2. Then we have $m\colon B \to C$ with $f = m \cdot e$. To prove that m is monic, consider u_1, $u_2\colon G \to B$ for $G \in \mathcal{G}$: if $mu_1 = mu_2$, we prove $u_1 = u_2$. Since G is regularly projective, there exist morphisms $u_i'\colon G \to A$ with $u_i = e \cdot u_i'$. From $fu_1' = fu_2'$ we conclude that there is $u\colon U \to P$ with $u_i' = p_i u$. Therefore $u_1 = ep_1 u = ep_2 u = u_2$.

(2) The functor $U\colon \mathcal{K} \to \mathbf{Set}^{\mathcal{G}}$ with components $\mathcal{K}(G, -)$ preserves and reflects both regular epimorphisms and isomorphisms. Preservation is clear. Let $f\colon A \to B$ be such that Uf is epic, i.e., sort-wise surjective. Factorize $f = m \cdot e$ where $e\colon A \to C$ is a regular epimorphism and $m\colon C \to B$ is a monomorphism. Then m is invertible because for every $G \in \mathcal{G}$ we see that all morphisms $G \to B$ factorize through m. Analogously, U reflects isomorphisms.

(3) Consequently, U preserves and reflects equivalences, see Remark 17(7).

▶ **Proposition 25.** *Let $\mathcal{K}$ be a category with kernel pairs and reflexive coequalizers, having a regular generator $\mathcal{G}$ formed by regular projectives. Equivalent are:*
(1) *all objects of $\mathcal{G}$ are effective, and*
(2) *$\mathcal{K}$ has effective equivalence relations.*

Proof. $2 \to 1$. Given an equivalence relations $e_1, e_2 \colon E \to A$ and its coequalizer $c \colon A \to B$, we prove that $\mathcal{K}(G, -)$ preserves it for every $G \in \mathcal{G}$. Let $f \colon \mathcal{K}(G, A) \to X$ be a function with $f \cdot \mathcal{K}(G, e_1) = f \cdot \mathcal{K}(G, e_2)$. We prove that it factorizes through $\mathcal{K}(G, c)$. In other words: for every pair $x_1, x_2 \in \mathcal{K}(G, A)$ merged by $\mathcal{K}(G, c)$ we prove that f also merges it. By assumption, $c \cdot x_1 = c \cdot x_2$. Thus x_1, x_2 factorize through the kernel pair of c, which by (2) is e_1, e_2. Given $h \colon G \to B$ with $x_i = e_i \cdot h$, we get

$$f(x_i) = f(e_i \cdot h) = \big(f \cdot \mathcal{K}(G, e_i)\big)(h) \qquad (i = 1, 2)\,.$$

This proves $f(x_1) = f(x_2)$.

$1 \to 2$. Let $\mathcal{G}$ consist of effective objects. For every equivalence relation e_1, $e_2 \colon E \to A$ form its coequalizer $c \colon A \to B$ and a kernel pair e_1', $e_2' \colon E' \to A$ of c. We have a factorization $h \colon E \to E'$ with $e_k = e_k' \cdot h$ for $k = 1, 2$. Our task is to prove that h is an isomorphism.

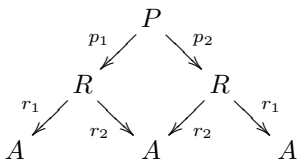
The functor $U = \mathcal{K}(G, -)$ for $G \in \mathcal{G}$ preserves limits and regular epimorphisms since G is a regular projective. Consequently, by Remarks 24 and 17(7) it preserves equivalence relations. Thus UE and UE' are equivalence relations on the set UA and, since G is effective, they have a common coequalizer Uc. It follows for the factorization morphism h that Uh is an isomorphism. Consequently, so is h. ◀

▶ Remark 26. The above proposition was inspired by the paper [11] in which an object is called an *effective projective* if its hom-functor preserves reflexive coequalizers. Preservation of coequalizers of equivalence relations seems a more suitable condition first because of the above proposition. Second, for varieties of infinitary algebras free algebras are effective but no longer effective projectives.

▶ **Definition 27.** *By a split coequalizer of morphisms r_1, $r_2 \colon R \to A$ is meant a morphism $c \colon A \to C$ with $cr_1 = cr_2$ such that there exist morphisms $i \colon C \to A$ and $j \colon A \to R$ with $ci = \mathrm{id}_C$, $jr_1 = \mathrm{id}_A$ and $jr_2 = ic$.*

▶ **Lemma 28.** *In a category with finite limits and regular factorizations for every split coequalizer c of a relation r_1, $r_2 \colon R \to A$ the kernel pair of c is the composite relation $R \circ R^0$.*

Proof. The relation R^0 is represented by r_2, $r_1 \colon R \to A$. Thus $R \circ R^0$ is represented by $r_1 p_1$, $r_1 p_2 \colon P \to A$ where p_1, $p_2 \colon P \to R$ is the kernel pair of r_2, see Remark 17(3):



The corresponding collectively monic pair r_1', $r_2' \colon R' \to A$ is obtained by factorizing $\langle r_1 p_1, r_1 p_2 \rangle \colon P \to A \times A$ as a regular epimorphism $e \colon P \to R'$ followed by a monomorphism $\langle r_1', r_2' \rangle$. We prove that r_1', r_2' is a kernel pair of c.

(1) $cr_1' = cr_2'$. Since e is epic, this follows from

$$c \cdot (r_1' e) = c \cdot (r_1 p_1) = cr_2 p_1 = cr_2 p_2 = c \cdot (r_1 p_2) = c \cdot (r_2' e)\,.$$

(2) Given $u_1, u_2 \colon U \to A$ with $cu_1 = cu_2$, then this pair uniquely factorizes through r_1', r_2'. Unicity is clear since $\langle r_1', r_2' \rangle$ is monic.

For i and j as in Definition 27 we see that

$$r_2 \cdot (ju_1) = icu_1 = icu_2 = r_2 \cdot (ju_2)\,.$$

Since p_1, p_2 is the kernel pair of r_2, this implies that there exists $h \colon U \to P$ with $ju_1 = p_1 h$ and $ju_2 = p_2 h$. The desired factorization is $eh \colon U \to R'$: we have

$$u_1 = (r_1 j)u_1 = r_1 p_1 h = r_1'(eh)$$

analogously $u_2 = r_2'(eh)$. ◀

▶ Remark 29. Given a reflexive relation $r_1, r_2 \colon R \to A$ in a category with finite limits and regular factorizations, then the relations R and $R \circ R^0$ have the same coequalizers. In fact, a morphism f merges the projections of R iff if merges those of $R \circ R^0$. Indeed, let $p_1, p_2 \colon P \to A$ be the kernel pair of f.

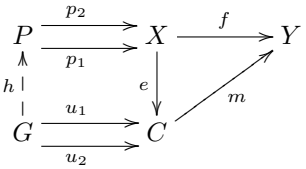
(1) If f merges R, then $R \subseteq P$ and since P is an equivalence relation, we conclude $R \circ R^0 \subseteq P \circ P^0 = P$.

(2) If f merges $R \circ R^0$, then since R is reflexive we have $\Delta_A \subseteq R^0$, hence, $R = R \circ \Delta_A \subseteq R \circ R^0$. Thus f merges r_1, r_2.

▶ **Proposition 30.** *Let $\mathcal{K}$ be a category with kernel pairs and their coequalizers. Then every strong generator consisting of effective objects is a regular generator.*

Proof.

(1) $\mathcal{K}$ has regular factorizations. Indeed, given a morphism $f \colon X \to Y$ form its kernel pair $p_1, p_2 \colon P \to X$ and a coequalizer $e \colon X \to C$ of p_1, p_2. We have a unique morphism $m \colon C \to Y$ with $f = m \cdot e$, and our task is to prove that m is monic. Since $\mathcal{G}$ is a strong generator, this is equivalent to proving for every pair $u_1, u_2 \colon G \to C$ with $G \in \mathcal{G}$ that $m \cdot u_1 = m \cdot u_2$ implies $u_1 = u_2$.



Since G is regularly projective, $\mathcal{K}(G, -)$ preserves regular epimorphisms. Thus there exist morphisms $v_i \colon G \to X$ with $u_i = e \cdot v_i$ $(i = 1, 2)$. From $m \cdot u_1 = m \cdot v_1$, we derive $f \cdot v_1 = f \cdot v_2$. Therefore, the pair v_1, v_2 factorizes through the kernel pair via a morphism $h \colon G \to P$ with $v_i = p_i \cdot h$. Thus $u_i = e \cdot p_i \cdot h$ and $e \cdot p_1 = e \cdot p_2$ implies $u_1 = u_2$.

(2) Consequently, every extremal epimorphism e is regular: given its regular factorization $e = m \cdot k$, we conclude that m is an isomorphism. Given a strong generator $\mathcal{G}$ consisting of effective objects, we prove that the canonical morphism c_X (Remark 9) is a regular epimorphism for every object X. We have an extremal epimorphism $e = [e_i] \colon \coprod_{i \in I} G_i \to X$ for some collection of objects $G_i \in \mathcal{G}$. Define $h \colon \coprod_{i \in I} G_i \to \coprod_{G \in \mathcal{G}} \coprod_{f \colon G \to X}$ to have the i-th component equal to the coproduct injection of $e_i \colon G_i \to X$. Then clearly $e = c_X \cdot h$. Since e is an extremal epimorphism, so is c_X. Thus c_X is a regular epimorphism. ◀

▶ **Corollary 31.** *Let $\mathcal{K}$ have an abstractly finite strong generator consisting of regular projectives. If $\mathcal{K}$ has reflexive coequalizers and kernel pairs, then it is complete and cocomplete.*

This follows from the above Proposition and Corollary 15.

4 A characterization of varieties

▶ **Definition 32** ([3]). *An endofunctor F of $\mathbf{Set}^S$ is called finitely bounded if for every object X and every element $x \in FX$ there exists a finite subobject $m\colon M \hookrightarrow X$ (that is, $\coprod_{s\in S} M_s$ is finite) with x in the image of Fm.*

▶ **Proposition 33** ([3]). *If S is a finite set, then an endofunctor of $\mathbf{Set}^S$ is finitely bounded iff it is finitary (i.e., preserves directed colimits).*

▶ **Example 34.** A finitely bounded endofunctor of $\mathbf{Set}^{\mathbb{N}}$ need not be finitary. Consider F assigning to X itself if all but finitely many sorts of X are empty, else FX is the terminal object. F is finitely bounded: given $x \in X$ of sort n, let $M \subseteq X$ have all sorts but n empty and $M_n = \{x\}$. Then $x \in FM_n$. But F does not preserves the colimit of the ω-chain of $\mathbb{N}$-sorted sets $X^{(k)} = (X^{(k)})_{n\in\mathbb{N}}$ for $k < \omega$ where $X^{(k)}_n = \emptyset$ if $k > n$, else $X^{(k)}_n = \{0, 1\}$.

The following proof is based on the idea of the proof of Theorem 4.4.5 in [7].

▶ **Theorem 35.** *A category is equivalent to a variety of finitary many-sorted algebras of finitely many sorts iff it has*
(a) *kernel pairs and reflexive coequalizers, and*
(b) *an abstractly finite, strong generator consisting of finitely many effective objects.*

▶ **Remark.** *If in (b) we require the generator to be regular (rather than just strong), then the assumption that kernel pairs exist can be dropped. See Corollary 15.*

Proof. Necessity follows from Remark 4(5) and Examples 21 and 10(2). To prove sufficiency, let $\{G_s\}_{s\in S}$ be a set as in (b) above in $\mathcal{K}$. From Corollary 15 and Remark 23(1) we know that $\mathcal{K}$ is complete and cocomplete. It has regular factorizations by Remark 24.

(i) The functor $U\colon \mathcal{K} \to \mathbf{Set}^S$ with components $\mathcal{K}(G_s, -)$ for $s \in S$ has a left adjoint $F\colon \mathbf{Set}^S \to \mathcal{K}$ with

$$FX = \coprod_{s\in S} X_s \bullet G_s \qquad \text{for} \quad X = (X_s)_{s\in S}\,.$$

Denote by $\mathbb{T}$ the corresponding monad on $\mathbf{Set}^S$. It is finitely bounded. Indeed, to give, for an S-sorted set X, an element of sort t in $TX = UFX$ means to give a morphism $f\colon G_t \to \coprod_{s\in S} X_s \bullet G_s$. Since $\{G_s\}_{s\in S}$ is abstractly finite, there is a finite sorted subset $m\colon M \hookrightarrow X$ such that f factorizes through the subcoproduct $FM \hookrightarrow FX$. That is, $f \in Tm[TM]$.
Since by the above proposition T is finitary, $\mathbf{Set}^{\mathbb{T}}$ is equivalent to a variety, see Theorem A40 in [5]. Thus it is sufficient to prove that U is monadic, then $\mathcal{K}$ is also equivalent to that variety.

(ii) To prove that U is monadic we use Beck's Theorem as formulated in [6] Theorem 3.3.10: U is monadic iff (1) U has a left adjoint, (2) U reflects isomorphisms, and (3) for every reflexive pair $r_1, r_2\colon R \to A$ in $\mathcal{K}$ if Ur_1, Ur_2 have a split coequalizer, then r_1, r_2 have a coequalizer and U preserves it.

(1) The left adjoint is given by $(X_s)_{s\in S} \mapsto \coprod_{s\in S} X_s \bullet G_s$.

(2) See Remark 24.

(3) Assuming first that r_1, r_2 are collectively monic, we work with the relation R on A. Since U preserves finite limits and reflects isomorphisms, it reflects finite limits. By Remark 24 $\mathcal{K}$ has regular factorizations and U preserves and reflects regular epimorphisms, relation composition and equivalence relations. Indeed, each G_s is regularly projective (Remark 23).

By assumption the relation $\bar{R}$ on UA represented by Ur_1, Ur_2 has a split coequalizer. Thus $\bar{R} \circ \bar{R}^0$ is an equivalence relation: see Lemma 28. Consequently, $R \circ R^0$ is an equivalence relation on A. Let $k\colon A \to K$ be its coequalizer. Then k is, since R is a reflexive relation, the coequalizer of R (Remark 29).

Since each G_s is effective, U preserves coequalizers of equivalence relations. Thus Uk is a coequalizer of $\bar{R} \circ \bar{R}^0$. Arguing as above, we conclude that Uk is also coequalizer of $U\bar{R}$. This proves that U preserves the coequalizer of r_1, r_2.

Let us next consider r_1, r_2 arbitrary. For Ur_1, Ur_2 we have a split coequalizer $c\colon UA \to C$ (Remark 27): there are morphisms $i\colon C \to UA$ and $j\colon UA \to UR$ satisfying $c \cdot i = \mathrm{id}_C$, $(Ur_1) \cdot j = \mathrm{id}_{UA}$ and $(Ur_2) \cdot j = i \cdot c$. Since $\mathcal{K}$ has finite products and regular factorizations, we can factorize r_1, r_2 as a regular epimorphism $e\colon R \to R'$ followed by a collectively monic pair r_1', $r_2'\colon R' \to A$. The relation R' is reflexive: given a morphism d with $r_i d = \mathrm{id}_A$, the morphism ed fulfils $r_i' ed = \mathrm{id}_A$. Moreover, the morphisms Ur_1', Ur_2' also have the split coequalizer c : the splitting is given by i and $Ue \cdot j$. Thus r_1', r_2' have a coequalizer preserved by U. Since e is epic, the coequalizers of r_1, r_2 and r_1', r_2' coincide; analogously for Ue: this is an epimorphism since each G_s is regularly projective. This concludes the proof. ◀

▶ **Corollary 36.** *A category is equivalent to a single-sorted variety iff it has*
(a) kernel pairs and reflexive coequalizers, and
(b) an effective, abstractly finite singleton strong generator.

▶ **Example 37.** None of the assumptions on the generator G in the above corollary can be omitted:

(1) Abstract finiteness. The one-element space G in the category of compact Hausdorff spaces forms a regular generator and is effective. But not abstractly finite: the copower $\coprod_{\mathbb{N}} G$ is the space $\beta\mathbb{N}$ and almost no morphism from G to $\beta\mathbb{N}$ factorizes through a finite subcopower. Analogously, no nonempty space is abstractly finite. Thus, the above category is not equivalent to a (finitary) variety.

(2) Effectivity. In the category DCPO (Example 3) consider the three-element chain 3. This is an abstractly finite strong generator (see [8]). But it is not effective: DCPO does not have effective equivalence relations.

(3) Strength. The one-element poset forms an abstractly finite, effective generator of **Pos**. But not a strong one. **Pos** also fails to have effective equivalence relations.

(4) Existence of copowers. Let $\mathbf{Set}_0$ be the full subcategory of **Set** on all nonempty sets. Here the terminal object 1 has almost all the required properties: every object is a copower of 1, every morphism from 1 to a coproduct factorizes through a coproduct injection, and 1 is effective. And 1 has "almost all" copowers – but not the empty one! Moreover, $\mathbf{Set}_0$ is clearly not equivalent to a variety: it is not complete.

▶ **Theorem 38.** *A category is equivalent to a variety of finitary, many-sorted algebras iff it has*

(a) *kernel pairs and reflexive coequalizers, and*

(b) *a strong generator consisting of finitely presentable effective objects.*

The proof is the same as that of the preceding theorem, except that the verification that T is finitary can be left out: since each G_s is finitely presentable, U preserves directed colimits, hence, so does $T = UF$.

▶ **Example 39.** Theorem 35 does not generalize to varieties with infinitely many sorts. We present a category $\mathcal{K}$ whose only finitely presentable object is the initial one (thus, $\mathcal{K}$ is not equivalent to a variety). Yet, $\mathcal{K}$ has coequalizers and an abstractly finite regular generator formed by effective objects.

$\mathcal{K}$ is the full subcategory of $\mathbf{Set}^{\mathbb{N}}$ consisting of the terminal object $\mathbb{1} = (1, 1, 1 \dots)$ and all objects $(X_n)_{n\in\mathbb{N}}$ such that for some $k \in \mathbb{N}$ we have

$$X_n \neq \emptyset \quad \text{iff} \quad n < k\,.$$

$\mathcal{K}$ is closed under coequalizers in $\mathbf{Set}^{\mathbb{N}}$. But not under colimits of ω-chains: consider the chain of inclusions of $A^{(k)} = (A^{(k)}_n)$ $(k < \omega)$ where $A^{(k)}_n = \{0, 1\}$ for $n \leq k$, else $\emptyset$. Then $colim_{k<\omega} A^{(k)} = \mathbb{1}$ in $\mathcal{K}$. And the only object of $\mathcal{K}$ whose hom-functor preserves this colimit is $(\emptyset, \emptyset, \emptyset \dots)$.

However, $\mathcal{K}$ has the abstractly finite regular generator $\{B^{(k)}\}_{k\in\mathbb{N}}$ where $B^{(k)}_n = 1$ for $n \leq k$, else $\emptyset$. Every morphism $f\colon B^{(k)} \to \coprod_{i\in I} B^{(k_i)}$ has the property that $k \leq k_i$ for some i, thus f factorizes through the coproduct injection of $B^{(k_i)}$. The verification that each $B^{(k)}$ is effective and that they form a regular generator is easy.

▶ Remark 40. Many-sorted varieties have also been characterized in [5] as precisely the strongly locally finitely presentable categories. We recall this shortly and show how this follows from the above reults.

(1) Using [4] Theorem 1.11, a *locally finitely presentable* category can be defined as a cocomplete category with a strong generator formed by finitely presentable objects.

(2) Let us recall that a small category is filtered iff colimits in **Set** with that domain commute with finite limits. Analogously one defines a small category to be *sifted* iff colimits in **Set** with that domain commute with finite products. An object A of a category $\mathcal{K}$ is called *perfectly presentable* if its hom-functor preserves sifted colimits (which are colimits of diagrams with sifted domains). If $\mathcal{K}$ has finite coproducts, these are precisely the finitely presentable objects A with $\mathcal{K}(A, -)$ preserving reflexive coequalizers ([5], Thm. 7.7). For example, in every variety all free algebras on finitely many generators are perfectly presentable ([5], Corollary 5.14).

(3) Using [5] Theorem 7.7, a *strongly locally finitely presentable* category can be defined as a cocomplete category with a strong generator $\mathcal{G}$ formed by perfectly presentable objects.

▶ **Corollary 41** ([5] Thm. 6.9)**.** *A category is strongly locally finitely presentable iff it is equivalent to a finitary many-sorted variety.*

Proof.

(1) Let $\mathcal{K}$ be strongly locally finitely presentable. For the strong generator $\mathcal{G}$ in the above remark every member $G \in \mathcal{G}$ is finitely presentable and effective, thus $\mathcal{G}$ is abstractly finite. By Theorem 38 $\mathcal{K}$ is equivalent to a variety.

(2) Conversely, every variety is strongly finitely presentable since the regular generator of Example 10(2) consists of perfectly presentable objects. ◀

Conclusions and Open Problems

Lawvere's characterization of (single-sorted) finitary varieties can be sharpened as follows: these are precisely the categories with reflexive coequalizers, kernel pairs and a strong generator formed by an abstractly finite effective object. We have presented a proof based on the theory of monads. And we have proved that for varieties of many-sorted algebras a completely analogous result holds in case of finitely many sorts: many-sorted varieties are, up to equivalence, precisely the categories with reflexive coequalizers, kernel pairs and an abstractly finite strong generator formed by finitely many effective objects. For infinitely many sorts a somewhat weaker characterization holds: instead of abstract finiteness one requires that the given generator consists of finitely presentable objects.

It is interesting to consider other base categories than **Set** or $\mathbf{Set}^S$. For example **Pos**: can one characterize (finitary) varieties of ordered algebras in a similar way?

Another direction of research are non-finitary varieties: what is an abstract characterization of categories monadic over $\mathbf{Set}^S$, or over **Pos**?

References

1 J. Adámek. A categorical characterization of varieties. *Algebra Universalis*, 51:215–234, 2004.
2 J. Adámek. On quasivarieties and varieties as categories. *Studia Logica*, 78:7–33, 2004.
3 J. Adámek, S. Milius, L. Sousa, and T. Wissmann. On finitary functors. *Theory Appl. Categories*, 34:1134–1164, 2019.
4 J. Adámek and J. Rosický. *Locally presentable and accessible categories*. Cambridge University Press, 1994.
5 J. Adámek, J. Rosický, and E. Vitale. *Algebraic Theories*. Cambridge University Press, 2011.
6 M. Barr and C. Well. *Toposes, Triples and Theories*. Springer Verlag, New York, 1985.
7 F. Borceux. *Handbook of categorical algebra*, volume 2. Cambridge Univ. Press, 1994.
8 J. Jurka and J. Rosický. Are chain complete posets co-wellpowered? to appear in Order. `arXiv:2007.02255`.
9 F.W. Lawvere. *Functorial semantics of algebraic theories*. Dissertation, Columbia Univ., 1963.
10 S. MacLane. *Categories for the working mathematician*. Springer, New York, 1997.
11 M.C. Pedicchio and R.J. Wood. A note on effectively projective objects. *J. Pure Appl. Algebra*, 158:83–87, 2001.

Tensor of Quantitative Equational Theories

Giorgio Bacci
Department of Computer Science, Aalborg University, Denmark

Radu Mardare
Department of Computer & Information Sciences, University of Strathclyde, Glasgow, UK

Prakash Panangaden
School of Computer Science, McGill University, Montreal, Canada

Gordon Plotkin
LFCS, School of Informatics, University of Edinburgh, UK

Abstract

We develop a theory for the commutative combination of quantitative effects, their *tensor*, given as a combination of quantitative equational theories that imposes mutual commutation of the operations from each theory. As such, it extends the sum of two theories, which is just their unrestrained combination. Tensors of theories arise in several contexts; in particular, in the semantics of programming languages, the monad transformer for global state is given by a tensor.

We show that under certain assumptions on the quantitative theories the free monad that arises from the tensor of two theories is the categorical tensor of the free monads on the theories. As an application, we provide the first algebraic axiomatizations of labelled Markov processes and Markov decision processes. Apart from the intrinsic interest in the axiomatizations, it is pleasing they are obtained compositionally by means of the sum and tensor of simpler quantitative equational theories.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting

Keywords and phrases Quantitative equational theories, Tensor, Monads, Quantitative Effects

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.7

1 Introduction

The theory of computational effects began with the work of Moggi [24, 25] seeking a unified category-theoretic account of the semantics of higher-order programming languages. He modelled computational effects (which he called notions of computation) by means of strong monads on a base category with cartesian closed structure. With Cenciarelli [5], he later extended the theory by allowing a compositional treatment of various semantic phenomena such as state, IO, exceptions, resumptions, etc, via the use of monad transformers. This work was followed up by the program of Plotkin and Power [26, 27] on an axiomatic understanding of computational effects as arising from operations and equations via the use of Lawvere theories (see also [14]). In a fundamental contribution [12] together with Hyland they developed a unified modular theory for algebraic effects that supports their combination by taking the *sum* and *tensor* of their Lawvere theories. This allowed them to recover in a more pleasing algebraic structural way many of the monad transformers considered by Moggi.

Quantitative equational theories, introduced by Mardare et al. [21], are a logical framework generalising the standard concept of equational logic to account for a concept of approximate equality. They are used to describe quantitative effects as monads on categories of metric spaces. Following the work of Hyland et al. [12], in [1] we developed a theory for the sum of quantitative equational theories, and proved that it corresponds to the categorical sum of quantitative effects as monads. As a major example, we axiomatised Markov processes with discounted probabilistic bisimilarity distance [7] as the sum of two theories: interpolative barycentric algebras (which axiomatises probability distributions with the Kantorovich metric [21]) and contractive operators (used to express the transition to the next state).

Whereas the sum of two monads is the simplest combination supporting both given effects with no interactions between them, the tensor additionally requires commutation of these effects over each other. Some of the most important monad transformers have an elegant abstract description using tensor. Specifically, Moggi's transformers for state, reader, and writer are examples of tensors [12].

In the present paper we extend the work initiated in [1], and develop the theory for the *tensor of quantitative equational theories*. The main contributions are:

1. we prove that the tensor of quantitative theories corresponds to the categorical tensor of their induced quantitative effects as strong monads;
2. we give quantitative axiomatisations to the *quantitative reader* and *writer monads*, from which we obtain analogues of Moggi's transformers at the level of theories using tensor;
3. we provide the first axiomatization of *labelled Markov processes* and *Markov decision processes* with their discounted bisimilarity metrics.

For the proof of (1) we introduce the concept of *pre-operation of a strong functor*, which we use to conveniently characterise the commutative bialgebras for the monads (which correspond to the Eilenberg-Moore algebras for their tensor). Crucially, this allows us to carry out the technical development directly at the level of quantitative equational theories without passing via a correspondence with metric-enriched Lawvere theories.

The axiomatisations in (3) are two major examples for our compositional theory quantitative effects. Specifically, we obtain the discounted bisimilarity metrics for labelled Markov processes and Markov decision processes with rewards by complementing the axiomatization for Markov processes presented in [1]. We model reactions to action labels by tensoring with the theory of quantitative reading computations (corresponding to Moggi's reader monad transformer); while rewards are recovered by tensoring with the theory of quantitative writing computations (corresponding to Moggi's writer monad transformer). We will illustrate our compositional approach by decomposing the proposed axiomatisations into their basic components and showing how to combine them step-by-step to get the desired result.

Further Related Work. In [12, 11] the tensor of (enriched) Lawvere theories is characterized as the colimit of certain commutative cocones, and the correspondence with the tensor of monads is obtained via the equivalence between Lawvere theories and monads. Since it is not hard to show that (basic) quantitative equational theories can be characterised as metric-enriched Lawvere theories, one may think to recover the correspondence with the tensor of monads via the equivalence with Lawvere theories. Alas, quantitative equational theories and Lawvere theories are not equivalent, as the latter allows generic operations with metric spaces as arities, while the framework of Mardare et al. [21] does not. An equivalence with *discrete* Lawvere theories [13] (where arities are just countable ordinals) does not hold either, because quantitative equations implicitly impose the existence operations with non-discrete arities which cannot be expressed in the framework of discrete Lawvere theories.

The above arguments required us to follow a different path, which led us to the introduction of pre-operations for a strong functor F. Pre-operations are related to Plokin and Power's *algebraic operations* [28, 29] in the sense that their assignment to F-algebras are the appropriate version of algebraic operations for functors. Moreover, when considered over a strong monad T they correspond to generic effects of type $I \to Tv$ (*i.e.*, Kleisli maps of type $I \to v$, where I is the identity for the monoidal product). The reason why we consider pre-operations over functors, and not just monads, is to relate the operations of an algebraic monad with those of its signature. This was crucial in the technical development of Section 5.

Finally, we remark that quantitative equational theories are a natural kind of enriched equational theory expressive enough to recover many examples of interest in computer science (see [21, 1, 23]), but not corresponding to metric-enriched Lawvere theories. In this respect, it is nice that also this simpler subclass of enriched theories are closed under sum and tensor.

2 Preliminaries and Notation

An *extended metric space* is a pair (X, d) consisting of a set X equipped with a distance function $d\colon X \times X \to [0, \infty]$ allowed to have infinite values, satisfying: (i) $d(x, y) = 0$ iff $x = y$, (ii) $d(x, y) = d(y, x)$ and (iii) $d(x, z) \leq d(x, y) + d(y, z)$.

A sequence (x_i) in X is *Cauchy* if $\forall \epsilon > 0, \exists N, \forall i, j \geq N, d(x_i, x_j) \leq \epsilon$. If every Cauchy sequence converges, the extended metric space (X, d) is said to be *complete*. If a space is not complete it can be completed by a well-known construction called *Cauchy completion*. We write $\overline{(X, d)}$, or just $\overline{X}$, for the completion of (X, d).

We denote by **Met** the category of extended metric spaces with morphisms the non-expansive maps, *i.e.* the $f\colon (X, d_X) \to (Y, d_Y)$ such that $d_X(x, y) \geq d_Y(f(x), f(y))$. This category is both complete (*i.e.*, have all limits) and cocomplete (*i.e.*, have all colimits). We will consider also the full subcategory **CMet** of complete extended metric spaces.

The categorical properties of extended metric spaces are much nicer than usual metric spaces. In particular, we note that **Met** is a symmetric monoidal category, with monoidal product $(X, d_X) \,\square\, (Y, d_Y)$ being the extended metric space with underlying set $X \times Y$ and extended metric $d_{X \square Y}((x, y)(x', y')) = d_X(x, x') + d_Y(y, y')$ (*cf.* [19]). Note that this is not the cartesian product in **Met** (for which $+$ above would be replaced by max).

The monoidal product $\square$ introduced above defines a closed monoidal structure on **Met**, with internal hom $[(X, d_X), (Y, d_Y)]$ given by the set of non-expansive maps from X to Y with point-wise supremum extended metric $d_{[X,Y]}(f, g) = \sup_{x \in X} d(f(x), g(x))$.

Finally, we recall the basic definitions of strong functor (and monad), strong natural transformations, and fix the notation (for more details see e.g. [17, 18]). Let $\mathbf{V}$ be a symmetric monoidal closed category with monoidal product[1] $\square\colon \mathbf{V} \times \mathbf{V} \to \mathbf{V}$, unit object $I \in \mathbf{V}$, and internal hom-functor $[-, -]\colon \mathbf{V} \times \mathbf{V} \to \mathbf{V}$. We will denote the *counit* (or evaluation map) of the adjunction $(V \,\square\, -) \dashv [V, -]$ by $ev^V\colon V \,\square\, [V, -] \Rightarrow Id$ and the *unit* (or co-evaluatation map) by $\overline{ev}^V\colon Id \to [V, V \,\square\, -]$.

A functor $F\colon \mathbf{V} \to \mathbf{V}$ is *strong* with *monoidal strength* $t_{V,W}\colon V \,\square\, F(W) \to F(V \,\square\, W)$, if t is a natural transformation satisfying the coherence conditions $F\lambda \circ t = \lambda$ and $t \circ (id \,\square\, t) \circ \alpha = F\alpha \circ t$, w.r.t. the associator α and left unitor λ of $\mathbf{V}$. The dual strength $\hat{t}_{V,W}\colon F(W) \,\square\, V \to F(W \,\square\, V)$ is given by $\hat{t} = F(s) \circ t \circ s$, where $s\colon V \,\square\, W \to W \,\square\, V$ is the natural isomorphism of the symmetric monoidal category $\mathbf{V}$. A natural transformation $\theta\colon F \Rightarrow G$ is said *strong* if F, G are strong functors with strengths t, σ, respectively, and $\sigma \circ (id \,\square\, \theta) = \theta \circ t$, meaning that θ interacts well with the strengths.

A monad (T, η, μ) with unit $\eta\colon Id \Rightarrow T$ and multiplication $\mu\colon TT \Rightarrow T$, is *strong* if T is a strong functor with strength t such that $t \circ (id \,\square\, \eta) = \eta$ and $\mu \circ tt = t \circ (id \,\square\, \mu)$.

Note that strong functors (resp. monads) on a symmetric monoidal closed category $\mathbf{V}$ are equivalent to $\mathbf{V}$-enriched functors (resp. monads) on the self-enriched category $\mathbf{V}$ [17].

3 Quantitative Equational Theories

Quantitative equations were introduced in [21]. In this framework equalities $t \equiv_\varepsilon s$ are indexed by a positive rational number, to capture the idea that t is "within ε" of s. This informal notion is formalised in a manner analogous to traditional equational logic. In this section we review this formalism.

[1] We denote the monoidal product by $\square$ to avoid confusion with other tensorial operations we will deal with in this paper, e.g., the tensor of monads.

Let Σ be a signature of function symbols $f\colon n \in \Sigma$ of arity $n \in \mathbb{N}$. Let X be a countable set of variables, ranged over by $x, y, z, \ldots$. We write $\mathbb{T}(\Sigma, X)$ for the set of Σ-terms freely generated over X, ranged over by $t, s, u, \ldots$.

A *substitution of type* Σ is a function $\sigma\colon X \to \mathbb{T}(\Sigma, X)$, canonically extended to terms as $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$; we write $\mathcal{S}(\Sigma)$ for the set of substitutions of type Σ.

A *quantitative equation of type* Σ over X is an expression of the form $t \equiv_\varepsilon s$, for $t, s \in \mathbb{T}(\Sigma, X)$ and $\varepsilon \in \mathbb{Q}_{\geq 0}$. We use $\mathcal{V}(\Sigma, X)$ to denote the set of quantitative equations of type Σ over X, and its subsets will be ranged over by $\Gamma, \Theta, \ldots$. Let $\mathcal{E}(\Sigma, X)$ be the set of *conditional quantitative equations* on $\mathbb{T}(\Sigma, X)$, which are expressions of the form

$$\{t_1 \equiv_{\varepsilon_1} s_1, \ldots, t_n \equiv_{\varepsilon_n} s_n\} \vdash t \equiv_\varepsilon s\,,$$

for arbitrary $s_i, t_i, s, t \in \mathbb{T}(\Sigma, X)$ and $\varepsilon_i, \varepsilon \in \mathbb{Q}_{\geq 0}$.

A *quantitative equational theory of type* Σ *over* X is a set $\mathcal{U}$ of conditional quantitative equations on $\mathbb{T}(\Sigma, X)$ containing the axioms and closed under the rules of inference below, for arbitrary $x, y, z, x_i, y_i \in X$, terms $s, t \in \mathbb{T}(\Sigma, X)$, rationals $\varepsilon, \varepsilon' \in \mathbb{Q}_{\geq 0}$, and $\Gamma, \Theta \subseteq \mathcal{V}(\Sigma, X)$,

(Refl) $\vdash x \equiv_0 x\,,$
(Symm) $\{x \equiv_\varepsilon y\} \vdash y \equiv_\varepsilon x\,,$
(Triang) $\{x \equiv_\varepsilon z, z \equiv_{\varepsilon'} y\} \vdash x \equiv_{\varepsilon+\varepsilon'} y\,,$
(Max) $\{x \equiv_\varepsilon y\} \vdash x \equiv_{\varepsilon+\varepsilon'} y\,,$ for all $\varepsilon' > 0\,,$
(Cont) $\{x \equiv_{\varepsilon'} y \mid \varepsilon' > \varepsilon\} \vdash x \equiv_\varepsilon y\,,$
(f-NE) $\{x_i \equiv_\varepsilon y_i \mid i{=}1..n\} \vdash f(x_1, \ldots, x_n) \equiv_\varepsilon f(y_1, \ldots, y_n)\,,$ for $f\colon n \in \Sigma\,,$
(Subst) If $\Gamma \vdash t \equiv_\varepsilon s$, then $\{\sigma(t) \equiv_\varepsilon \sigma(s) \mid t \equiv_\varepsilon s \in \Gamma\} \vdash \sigma(t) \equiv_\varepsilon \sigma(s)\,,$ for $\sigma \in \mathcal{S}(\Sigma)\,,$
(Ass) If $t \equiv_\varepsilon s \in \Gamma$, then $\Gamma \vdash t \equiv_\varepsilon s\,,$
(Cut) If $\Gamma \vdash \Theta$ and $\Theta \vdash t \equiv_\varepsilon s$, then $\Gamma \vdash t \equiv_\varepsilon s\,,$

where we write $\Gamma \vdash \Theta$ to mean that $\Gamma \vdash t \equiv_\varepsilon s$ holds for all $t \equiv_\varepsilon s \in \Theta$.

The rules (Subst), (Cut), (Ass) are the usual rules of equational logic. The axioms (Refl), (Symm), (Triang) correspond, respectively, to reflexivity, symmetry, and the triangle inequality; (Max) represents inclusion of neighborhoods of increasing diameter; (Cont) is the limiting property of a decreasing chain of neighborhoods with converging diameters; and (f-NE) expresses non-expansivness of $f \in \Sigma$.

A set A of conditional quantitative equations *axiomatises* a quantitative equational theory $\mathcal{U}$, if $\mathcal{U}$ is the smallest quantitative equational theory containing A.

The models of these theories, called *quantitative* Σ*-algebras*, are Σ-algebras in **Met**.

▶ **Definition 1** (Quantitative Algebra). *A* quantitative Σ-algebra *is a tuple* $\mathcal{A} = (A, \Sigma^{\mathcal{A}})$*, where* A *is an extended metric space and* $\Sigma^{\mathcal{A}} = \{f^{\mathcal{A}}\colon A^n \to A \mid f\colon n \in \Sigma\}$ *is a set of non-expansive* interpretations *(i.e., satisfying* $\max_i d_A(a_i, b_i) \geq d_A(f^{\mathcal{A}}(a_1, \ldots, a_n), f^{\mathcal{A}}(b_1, \ldots, b_n))$*).*

The morphisms between quantitative Σ-algebras are non-expansive Σ-homomorphisms. Quantitative Σ-algebras and their morphism form a category, denoted by $\mathbf{QA}(\Sigma)$.

$\mathcal{A} = (A, \Sigma^{\mathcal{A}})$ *satisfies* the conditional quantitative equation $\Gamma \vdash t \equiv_\varepsilon s$ in $\mathcal{E}(\Sigma, X)$, written $\Gamma \models_{\mathcal{A}} t \equiv_\varepsilon s$, if for any assignment $\iota\colon X \to A$, the following implication holds

$$\big(\forall t' \equiv_{\varepsilon'} s' \in \Gamma\,, d_A(\iota(t'), \iota(s')) \leq \varepsilon'\big) \Rightarrow d_A(\iota(t), \iota(s)) \leq \varepsilon\,,$$

where $\iota(t)$ is the homomorphic interpretation of t in $\mathcal{A}$.

A quantitative algebra $\mathcal{A}$ is said to *satisfy* (or be a *model* for) the quantitative theory $\mathcal{U}$, if $\Gamma \models_{\mathcal{A}} t \equiv_\varepsilon s$ whenever $\Gamma \vdash t \equiv_\varepsilon s \in \mathcal{U}$. We write $\mathbb{K}(\Sigma,\mathcal{U})$ for the collection of models of a theory $\mathcal{U}$ of type Σ.

Sometimes it is convenient to consider the quantitative Σ-algebras whose carrier is a complete extended metric space. This class of algebras forms a full subcategory of $\mathbf{QA}(\Sigma)$, written $\mathbf{CQA}(\Sigma)$. Similarly, we write $\mathbb{CK}(\Sigma,\mathcal{U})$ for the full subcategory of quantitative Σ-algebras in $\mathbf{CQA}(\Sigma)$ which are models of $\mathcal{U}$.

The following lifts the Cauchy completion of metric spaces to quantitative algebras.

▶ **Definition 2.** *(Algebra Completion) The* Cauchy completion of a quantitative Σ-algebra $\mathcal{A} = (A, \Sigma^{\mathcal{A}})$*, is the quantitative* Σ*-algebra* $\overline{\mathcal{A}} = (\overline{A}, \Sigma^{\overline{\mathcal{A}}})$*, where* $\overline{A}$ *is the Cauchy completion of* A *and* $\Sigma^{\overline{\mathcal{A}}} = \{f^{\overline{\mathcal{A}}} \colon \overline{A}^n \to \overline{A} \mid f \colon n \in \Sigma\}$ *is such that for Cauchy sequences* $(b^i_j)_j$ *converging to* $b^i \in \overline{A}$*, for* $1 \le i \le n$*,* $f^{\overline{\mathcal{A}}}(b^1, \dots, b^n) = \lim_j f^{\mathcal{A}}(b^1_j, \dots, b^n_j)$.

The above extends to a functor $\mathbb{C} \colon \mathbf{QA}(\Sigma) \to \mathbf{CQA}(\Sigma)$ which is the left adjoint to the functor embedding $\mathbf{CQA}(\Sigma)$ into $\mathbf{QA}(\Sigma)$.

The completion of quantitative Σ-algebras extends also to a functor from $\mathbb{K}(\Sigma,\mathcal{U})$ to $\mathbb{CK}(\Sigma,\mathcal{U})$, whenever $\mathcal{U}$ can be axiomatised by a collection of *continuous schemata*, which are conditional quantitative equations of the form

$$\{x_i \equiv_{\varepsilon_i} y_i \mid i = 1..n\} \vdash t \equiv_\varepsilon s\,, \qquad \text{for all } \varepsilon \ge f(\varepsilon_1, \dots, \varepsilon_n),$$

where $f \colon \mathbb{R}^n_{\ge 0} \to \mathbb{R}_{\ge 0}$ is a continuous real-valued function, $\varepsilon, \varepsilon_i \in \mathbb{Q}_{\ge 0}$, and $x_i, y_i \in X$. We call such a theory *continuous*.

Free Monads on Quantitative Equational Theories

To every signature Σ, one can associate a *signature endofunctor* (also called Σ) on **Met** by:

$$\Sigma = \coprod_{f : n \in \Sigma} Id^n\,.$$

It is easy to see that, by couniversality of the coproduct, quantitative Σ-algebras correspond to Σ-algebras for the functor Σ in **Met**, and the morphisms between them to non-expansive homomorphisms of Σ-algebras. Below we pass between the two points of view as convenient.

▶ **Theorem 3** (Free Algebra [21])**.** *The forgetful functor* $\mathbb{K}(\Sigma,\mathcal{U}) \to \mathbf{Met}$ *has a left adjoint.*

The left adjoint assigns to any $X \in \mathbf{Met}$ a *free quantitative* Σ*-algebra* $(T_X, \psi^{\mathcal{U}}_X)$ satisfying $\mathcal{U}$, from which one canonically obtains the monad $(T_{\mathcal{U}}, \eta^{\mathcal{U}}, \mu^{\mathcal{U}})$, with functor $T_{\mathcal{U}} \colon \mathbf{Met} \to \mathbf{Met}$ mapping $X \in \mathbf{Met}$ to the carrier T_X of the free algebra.

A similar free construction also holds for quantitative algebras in $\mathbf{CQA}(\Sigma)$ for continuous quantitative equational theories, implying that the forgetful functor from $\mathbb{CK}(\Sigma,\mathcal{U})$ to **CMet** has a left adjoint. In particular, $\mathbb{C}T_{\mathcal{U}}$ is the free monad on $\mathcal{U}$ in **CMet**, provided that the quantitative equational theory is continuous.

Finally, let T-**Alg** denote the category of Eilenberg-Moore (EM) algebras for a monad T. In [1], it is shown that, whenever the quantitative theory $\mathcal{U}$ is *basic*, *i.e.*, it can be axiomatised by a set of conditional equations of the form

$$\{x_1 \equiv_{\varepsilon_1} y_1, \dots, x_n \equiv_{\varepsilon_n} y_n\} \vdash t \equiv_\varepsilon s\,,$$

where $x_i, y_i \in X$ (*cf.* [22]), then EM $T_{\mathcal{U}}$-algebras are in 1-1 correspondence with the quantitative algebras satisfying $\mathcal{U}$:

▶ **Theorem 4.** *For any basic quantitative equational theory* $\mathcal{U}$ *of type* Σ*,* $T_{\mathcal{U}}$**-Alg** $\cong \mathbb{K}(\Sigma,\mathcal{U})$.

4 Tensor of Strong Monads

In this section we provide the definition of *tensor of strong monads* on a generic symmetric monoidal closed category $\mathbf{V}$. The presentation follows and generalises that of Manes [20], which considers only the case of monads on **Set**.

Let v be an object in $\mathbf{V}$. As $\mathbf{V}$ is self-enriched, it has all v-fold *powers* (or *v-powers*) X^v, of any object $X \in \mathbf{V}$, defined as $X^v = [v, X]$ [16]. Moreover, $(-)^v \colon \mathbf{V} \to \mathbf{V}$ is a strong functor with strength $\xi_{X,Y} \colon X \square Y^v \to (X \square Y)^v$ obtained by currying

$$v \square (X \square Y^v) \xrightarrow{\cong} X \square (v \square Y^v) \xrightarrow{X \square ev} X \square Y \,.$$

Let $F \colon \mathbf{V} \to \mathbf{V}$ be a strong functor with strength t. The *v-power* functor $(-)^v$ can be lifted to F-algebras by mapping (A, a) to $(A, a)^v = (A^v, a^v \circ \sigma_A)$, where $\sigma_A \colon FA^v \Rightarrow (FA)^v$ is the strong natural transformation obtained from t by currying $F ev^v_A \circ t_{v, A^v}$. Hence F-algebras are closed under powers of $\mathbf{V}$-objects.

▶ **Definition 5** (Pre-operation of a strong functor). *Let $F \colon \mathbf{V} \to \mathbf{V}$ be a strong functor and $v \in \mathbf{V}$. A v-*ary pre-operation of F *is a strong natural transformation of type $(-)^v \Rightarrow F$.*

We denote by $\mathcal{O}_F(v)$ the set of v-ary pre-operations of F. An application of $g \in \mathcal{O}_F(v)$ to an F-algebra (A, a) is the composite $a^g = a \circ g_A$. We call a^g an *operation* of (A, a).

▶ **Proposition 6.** *Let $(A, a), (B, b)$ be F-algebras of a strong endofunctor F on $\mathbf{V}$ and $f \colon A \to B$ a morphism in $\mathbf{V}$. Then, the following are equivalent:*
1. *f is an F-homomorphisms from (A, a) to (B, b);*
2. *For every $v \in \mathbf{V}$ and $g \in \mathcal{O}_F(v)$, $f \circ a^g = b^g \circ f^v$.*

The above proposition indicates that F-algebras are precisely characterised by their operations. In some situations, depending on the functor F, one gets the same characterisation with much fewer operations. We identify this property with the following definition.

▶ **Definition 7** (Density). *A set $\mathcal{D}$ of pre-operations of a strong functor $F \colon \mathbf{V} \to \mathbf{V}$ is* dense, *if for any F-algebras (A, a), (B, b) and $f \colon A \to B$ in $\mathbf{V}$, the following are equivalent:*
1. *f is an F-homomorphisms from (A, a) to (B, b);*
2. *For every v-ary pre-operation $g \in \mathcal{D}$, $f \circ a^g = b^g \circ f^v$.*

Let F, G be two strong endofunctors on $\mathbf{V}$. A *$\langle F, G \rangle$-bialgebra* is a triple (A, a, b) consisting of an object $A \in \mathbf{V}$ with both an F-algebra structure $a \colon FA \to A$ and a G-algebra structure $b \colon GA \to A$. A morphism of $\langle F, G \rangle$-bialgebras is an arrow that is simultaneously an F- and G-homomorphism. Denote by $\langle F, G \rangle$-**biAlg** the category of $\langle F, G \rangle$-bialgebras.

▶ **Proposition 8.** *Let (A, a, b) be an $\langle F, G \rangle$-bialgebra. The following statements are equivalent:*
1. *For all $v \in \mathbf{V}$ and $g \in \mathcal{O}_F(v)$, a^g is a G-homomorphism;*
2. *For all $w \in \mathbf{V}$ and $h \in \mathcal{O}_G(w)$, b^h is an F-homomorphism.*

Diagrammatically:

$$\begin{array}{ccc} GA^v & \xrightarrow{\bar{b}} & A^v \\ {\scriptstyle G(a^g)}\downarrow & (1) & \downarrow{\scriptstyle a^g} \\ GA & \xrightarrow{b} & A \end{array} \quad \textit{iff} \quad \begin{array}{ccc} FA^w & \xrightarrow{\bar{a}} & A^w \\ {\scriptstyle F(b^h)}\downarrow & (2) & \downarrow{\scriptstyle b^h} \\ FA & \xrightarrow{a} & A \end{array}$$

where $(A, a)^w = (A^w, \bar{a})$ and $(A, b)^v = (A^v, \bar{b})$.

▶ **Definition 9** (Commutative bialgebra). *A $\langle F, G\rangle$-bialgebra (A, a, b) is* commutative *if it satisfies either of the equivalent conditions of Proposition 8.*

In the case the functors F and G admit dense sets of pre-operations, commutativity for their bialgebras can be more conveniently expressed in the following way.

▶ **Proposition 10.** *Let $\mathcal{D}$ and $\mathcal{E}$ be dense sets of pre-operations for F and G, respectively. A $\langle F, G\rangle$-bialgebra (A, a, b) is* commutative *iff it satisfies either of the equivalent conditions:*
1. *For all $g \in \mathcal{D}$, a^g is a G-homomorphism;*
2. *For all $h \in \mathcal{E}$, b^h is an F-homomorphism.*

Let (T, η, μ) be a strong monad on $\mathbf{V}$. Note that, as T is a strong functor and the EM-algebras for T are closed under powers of $\mathbf{V}$-objects, all the results and definitions given in this section extends to EM-algebras for T.

Let (T, η, μ), (T', η', μ') be two strong monads on $\mathbf{V}$. A EM $\langle T, T'\rangle$-*bialgebra* is a triple (A, a, a') consisting of an object $A \in \mathbf{V}$ with both an EM T-algebra structure $a\colon TA \to A$ and an EM T'-algebra structure $a'\colon T'A \to A$. We say that an EM $\langle T, T'\rangle$-bialgebra (A, a, a') is *commutative* if it is so as a $\langle T, T'\rangle$-bialgebra for the functors T, T'. We denote by $\langle T, T'\rangle$-**biAlg** the category of EM $\langle T, T'\rangle$-bialgebras and by $(T \otimes T')$-**biAlg**, the full subcategory of the commutative EM $\langle T, T'\rangle$-bialgebras.

▶ **Definition 11** (Tensor of monads). *If the forgetful functor $(T \otimes T')$-**biAlg** $\to \mathbf{V}$ has a left adjoint, then the monad induced by the adjunction is the* tensor *of T, T', denoted $T \otimes T'$.*

Note that the tensor of monads does not necessarily exist (see [4] for counterexamples). However, when it does $T \otimes T' \cong T' \otimes T$, as the categories of commutative biagebras $(T \otimes T')$-**biAlg** and $(T' \otimes T)$-**biAlg** are isomorphic.

5 Tensor of Quantitative Theories

In this section, we develop the theory for the *tensor* of quantitative equational theories. The main result is that the free monad on the tensor of two theories is the tensor of the monads on the theories. In the proof given, we use the fact that the quantitative theories are *basic*, as this allows us to exploit the correspondence between the algebras of a theory $\mathcal{U}$ and the EM-algebras of the monad $T_{\mathcal{U}}$ (Theorem 4).

Let Σ, Σ' be two disjoint signatures. Following Freyd [8] (and [12]), we define the tensor of two quantitative equational theories $\mathcal{U}, \mathcal{U}'$ of respective types Σ and Σ', written $\mathcal{U} \otimes \mathcal{U}'$, as the smallest quantitative theory containing $\mathcal{U}, \mathcal{U}'$ and the quantitative equations

$$\vdash f(g(x_1^1, \dots, x_m^1), \dots, g(x_1^n, \dots, x_m^n)) \equiv_0 g(f(x_1^1, \dots, x_1^n), \dots, f(x_m^1, \dots, x_m^n)), \tag{1}$$

for all $f\colon n \in \Sigma$ and $g\colon m \in \Sigma'$, expressing that the operations of one theory commute with the operations of the other.

5.1 Density of Symbolic Pre-operations

Towards our main result, we identify a dense set of pre-operations for the free monads on quantitative equational theories which, in turn, will gives us a simpler characterization for commutative bialgebras for these monads (*cf.* Proposition 10).

First notice that any signature functor $\Sigma = \coprod_{f:n\in\Sigma} Id^n$ in **Met** is strong, as it is the coproduct of the strong functors $Id^n \cong (-)^{\underline{n}}$, where $\underline{n} \in$ **Met** denotes the set $\{1, \dots, n\}$

equipped with the discrete extended metric assigning infinite distance to distinct elements. Moreover, the injections $in_f \colon (-)^{\underline{n}} \Rightarrow \Sigma$ are strong natural transformations, hence they are $\underline{n}$-ary pre-operations of Σ (*cf.* Definition 5).

▶ **Proposition 12.** *$\mathcal{S}_\Sigma = \{in_f \mid f \colon n \in \Sigma\}$ is a dense set of pre-operations of Σ.*

In the following, the pre-operations in $\mathcal{S}_\Sigma$ will be called *symbolic*, and to simplify the notation, for any $f \colon n \in \Sigma$ and Σ-algebra (A, a), we write a^f instead of a^{in_f}.

Let $\mathcal{U}$ be a quantitative equational theory of type Σ. Then, also the monad $T_\mathcal{U}$ is strong, with strength $\zeta_{X,Y} \colon X \square T_\mathcal{U} Y \to T_\mathcal{U}(X \square Y)$ obtained by uncurrying the unique map $h_{X,Y}$ that, by Theorem 3, makes the following diagram commute

$$\begin{array}{ccccc} Y & \xrightarrow{\eta^\mathcal{U}_Y} & T_\mathcal{U} Y & \xleftarrow{\quad\psi^\mathcal{U}_Y\quad} & \Sigma T_\mathcal{U} Y \\ {\scriptstyle \beta_{X,Y}}\searrow & & \downarrow{\scriptstyle h_{X,Y}} & & \downarrow{\scriptstyle \Sigma h_{X,Y}} \\ & & (T_\mathcal{U}(X \square Y))^X & \xleftarrow{\psi^\mathcal{U}_{(T(X\square Y))^X}} & \Sigma(T_\mathcal{U}(X \square Y))^X \end{array}$$

where $\beta_{X,Y}$ is the currying of $\eta^\mathcal{U}_{X \square Y} \colon X \square Y \to T_\mathcal{U}(X \square Y)$.

Since a monad is strong iff both its unit and multiplication are strong natural transformations, both $\eta^\mathcal{U}$, $\mu^\mathcal{U}$ are strong. Moreover, also $\psi^\mathcal{U} \colon \Sigma T_\mathcal{U} \Rightarrow T_\mathcal{U}$ is strong.

Thus any pre-operation $g \in \mathcal{O}_\Sigma(v)$ can be tuned into an pre-operation of $T_\mathcal{U}$ as the composite

$$(-)^v \xrightarrow{g} \Sigma \xrightarrow{\Sigma\eta^\mathcal{U}} \Sigma T_\mathcal{U} \xrightarrow{\psi^\mathcal{U}} T_\mathcal{U} \,.$$

In particular, when the theory $\mathcal{U}$ is basic, by exploiting Theorem 4, the above transformation allows us to turn any dense set of pre-operations of Σ into a dense set of pre-operations of $T_\mathcal{U}$.

▶ **Proposition 13.** *Let $\mathcal{U}$ be a basic quantitative theory of type Σ and $\mathcal{D}$ a dense set of pre-operations of Σ. Then $\{\psi^\mathcal{U} \circ \Sigma\eta^\mathcal{U} \circ g \mid g \in \mathcal{D}\}$ is a dense set of pre-operations of $T_\mathcal{U}$.*

By combining Propositions 12 and 13, we have that $\mathcal{S}_{T_\mathcal{U}} = \{\psi^\mathcal{U} \circ \Sigma\eta^\mathcal{U} \circ in_f \mid f \colon n \in \Sigma\}$ is a dense set of pre-operations of $T_\mathcal{U}$. We call also these pre-operations *symbolic* and we simplify the notation by writing $a^{\langle f \rangle}$ instead of $a^{(\psi^\mathcal{U} \circ \Sigma\eta^\mathcal{U} \circ in_f)}$, for $f \colon n \in \Sigma$ and $(A, a) \in T_\mathcal{U}$-**Alg**.

Thus, as an immediate consequence of Propositions 10 and 13, we obtain the following simpler characterization for commutative $\langle T_\mathcal{U}, T_{\mathcal{U}'} \rangle$-bialgebras.

▶ **Corollary 14.** *Let $\mathcal{U}$, $\mathcal{U}'$ be basic quantitative theories respectively of type Σ, Σ'. A $\langle T_\mathcal{U}, T_{\mathcal{U}'} \rangle$-bialgebra (A, a, b) is commutative iff it satisfies either of the equivalent conditions*
1. *For all $f \colon n \in \Sigma$, $a^{\langle f \rangle}$ is a $T_{\mathcal{U}'}$-homomorphism;*
2. *For all $g \colon n \in \Sigma'$, $b^{\langle g \rangle}$ is a $T_\mathcal{U}$-homomorphism.*

5.2 Tensor of Free Monads on Quantitative Theories

Let $\mathcal{U}, \mathcal{U}'$ be basic quantitative theories respectively of type Σ, Σ'. We show that any model for $\mathcal{U} \otimes \mathcal{U}'$ is a *$\langle \mathcal{U} \otimes \mathcal{U}' \rangle$-bialgebra*: an extended metric space A with both a Σ-algebra structure $a \colon \Sigma A \to A$ satisfying $\mathcal{U}$ and a Σ'-algebra structure $b \colon \Sigma' A \to A$ satisfying $\mathcal{U}'$ and respecting the diagrammatic condition below, for all $f \colon n \in \Sigma$ and $g \colon m \in \Sigma'$

$$\begin{array}{ccccc} A^{\underline{n}} & \xrightarrow{a^f} & A & \xleftarrow{b^g} & A^{\underline{m}} \\ {\scriptstyle (b^g)^{\underline{n}}}\uparrow & & & & \uparrow{\scriptstyle (a^f)^{\underline{m}}} \\ (A^{\underline{m}})^{\underline{n}} & & \xrightarrow[\cong]{\chi} & & (A^{\underline{n}})^{\underline{m}} \end{array} \tag{2}$$

Formally, we denote by $(\mathcal{U}\otimes\mathcal{U}')$-**biAlg** the category of $\langle\mathcal{U}\otimes\mathcal{U}'\rangle$-bialgebras, with morphisms the non-expansive homomorphisms preserving both algebraic structures. Then, the following isomorphism of categories holds.

▶ **Proposition 15.** $\mathbb{K}(\Sigma+\Sigma',\mathcal{U}\otimes\mathcal{U}') \cong (\mathcal{U}\otimes\mathcal{U}')$-**biAlg**, *for* $\mathcal{U},\mathcal{U}'$ *basic quantitative theories.*

Moreover, by adapting the isomorphism of Theorem 4 and exploiting the density of symbolic pre-operations (*cf.* Corollary 14) the following is also true.

▶ **Proposition 16.** $(\mathcal{U}\otimes\mathcal{U}')$-**biAlg** $\cong (T_{\mathcal{U}}\otimes T_{\mathcal{U}'})$-**biAlg**, *for* $\mathcal{U},\mathcal{U}'$ *basic quantitative theories.*

By combining the above two propositions we get the main theorem of this section.

▶ **Theorem 17.** *Let* $\mathcal{U},\mathcal{U}'$ *be basic quantitative theories. Then, the monad* $T_{\mathcal{U}\otimes\mathcal{U}'}$ *in* **Met** *is the tensor of monads* $T_{\mathcal{U}}\otimes T_{\mathcal{U}'}$.

Proof. By Propositions 15 and 16 the forgetful functor from $(T_{\mathcal{U}}\otimes T_{\mathcal{U}'})$-**biAlg** to **Met** has a left adjoint and the monad generated by this adjunction is isomorphic to $T_{\mathcal{U}\otimes\mathcal{U}'}$. Thus, by definition of tensor of monads, $T_{\mathcal{U}\otimes\mathcal{U}'} \cong T_{\mathcal{U}}\otimes T_{\mathcal{U}'}$. ◀

The above results do not require any specific property of **Met**, apart that its morphisms are non-expansive maps. Thus, when the quantitative equational theories are continuous, we can reformulate an alternative version of Theorem 17 which is valid in **CMet**.

▶ **Theorem 18.** *Let* $\mathcal{U},\mathcal{U}'$ *be continuous quantitative theories. Then,* $\mathbb{C}T_{\mathcal{U}\otimes\mathcal{U}'}$ *in* **CMet** *is the tensor of monads* $\mathbb{C}T_{\mathcal{U}}\otimes\mathbb{C}T_{\mathcal{U}'}$.

6 Quantitative Reader Algebras

Let E be a finite set of input values and fix an enumeration $E=\{e_1,\dots,e_n\}$ for it. The *quantitative reader algebras* of type E are the algebras for the signature

$$\Sigma_{\mathcal{R}_E} = \{\mathsf{r}\colon |E|\}$$

having only one operator r of arity equal to the number of the input values in E, and satisfying the following axioms

$$(\mathsf{Idem}) \vdash x \equiv_0 \mathsf{r}(x,\dots,x)\,,$$
$$(\mathsf{Diag}) \vdash \mathsf{r}(x_{1,1},\dots,x_{n,n}) \equiv_0 \mathsf{r}(\mathsf{r}(x_{1,1},\dots,x_{1,n}),\dots,\mathsf{r}(x_{n,1},\dots,x_{n,n}))\,.$$

The quantitative theory induced by the axioms above, written $\mathcal{R}_E$, is called *quantitative theory of reading computations.*

Intuitively, the term $\mathsf{r}(t_1,\dots,t_n)$ can be interpreted as the computation that proceeds as t_i after reading the value e_i from its input. The axiom (Idem) says that if we ignore the value of the input the reading of it is not observable; (Diag) says that the resulting computation after reading the input is the same no matter how many times we read it.

▶ Remark 19. For the binary case ($|E|=2$) we can think of r as an *if-then-else* statement $b?(S,T)$ checking for the value of a fixed global Boolean variable b and proceeding as S when $b=$ true, and as T otherwise. In this case, (Idem) and (Diag) express the standard program equivalences $S\equiv b?(S,S)$ and $b?(S,T)\equiv b?(b?(S,T'),b?(S',T))$.

In the following, when the set E is clear from the context, we use $\mathcal{R}$ in place of $\mathcal{R}_E$.

On Metric Spaces

Let E be a finite set. We denote by $\underline{E}$ the extended metric space on E equipped with the indiscrete metric that assigns infinite distance to any pair of distinct elements.

Consider the $\underline{E}$-power functor $(-)^{\underline{E}}\colon \mathbf{Met} \to \mathbf{Met}$, assigning to each $X \in \mathbf{Met}$ the metric space $[E, X]$ of (necessarily non-expansive) maps from $\underline{E}$ to X.

This functor has a monad structure, with unit $\kappa\colon Id \Rightarrow (-)^{\underline{E}}$ and multiplication $\Delta\colon ((-)^{\underline{E}})^{\underline{E}} \Rightarrow (-)^{\underline{E}}$, respectively given as follows, for $x \in X$, $e \in E$, and $f \in E \to X^{\underline{E}}$

$$\kappa_X(x)(e) = x\,, \qquad\qquad \Delta_X(f)(e) = f(e)(e)\,.$$

This is also known as *reader monad* (also called *environment monad* or *function monad*).

▶ Remark 20. The reader monad is always well defined in a cartesian closed category. Fix an object E. The reader monad $(-)^E$ has unit and multiplication respectively given by

$$X \cong X^1 \xrightarrow{X^!} X^E \qquad \text{and} \qquad (X^E)^E \cong X^{E\times E} \xrightarrow{X^\delta} X^E\,,$$

where $!\colon E \to 1$ is the unique map to the terminal object and $\delta\colon E \to E \times E$ the diagonal map $\delta = \langle id, id\rangle$. However, this definition does not generalise to arbitrary monoidal closed categories, and **Met** is such a counterexample. The specific problem with **Met** is that $\delta\colon E \to E \,\square\, E$ is not well-defined for arbitrary $E \in \mathbf{Met}$, as non-expansivness requires that

$$d_E(e, e') \geq d_{E\square E}(\delta(e), \delta(e')) = d_E(e, e') + d_E(e, e')\,,$$

which holds only when E has the discrete metric. This is the reason why in our treatment we restrict the set of input values to have discrete metric.

The reader monad $(-)^{\underline{E}}$ is isomorphic to the free monad $T_{\mathcal{R}}$. In other words, the quantitative theory $\mathcal{R}$ of reading computations axiomatises the reader monad.

▶ **Theorem 21.** *The monads $T_{\mathcal{R}}$ and $(-)^{\underline{E}}$ in* **Met** *are isomorphic.*

Let T be a strong monad with strength t. The natural transformation $\lambda_X\colon TX^{\underline{E}} \Rightarrow (TX)^{\underline{E}}$ obtained from the strength t by currying $T ev_X^{\underline{E}} \circ t_{\underline{E}, X^{\underline{E}}}$, is a distributive law of monads. Distributive laws induce a notion of monad composition [2], so Moggi's reader monad transformer $T \mapsto (T-)^{\underline{E}}$ is also available in **Met**. The following says that we can recover this monad transformer as the operation of tensoring with the reader monad.

▶ **Theorem 22** (Tensoring with Reader Monad)**.** *Let T be a strong monad. Then, $T \otimes (-)^{\underline{E}}$ exists and is given as the monad composition $(T-)^{\underline{E}}$.*

By using the above result in combination with Theorem 17, we obtain an analogous transformer at the level of quantitative equational theories as follows.

▶ **Corollary 23.** *Let $\mathcal{U}$ be a basic quantitative equational theory. Then, $(T_{\mathcal{U}}-)^{\underline{E}}$ is the free monad on the theory $\mathcal{U} \otimes \mathcal{R}$ in* **Met**.

On Complete Metric Spaces

The category **CMet** has finite products. Since, we assumed the set of input values E to be finite, the functor $(-)^{\underline{E}}$ is isomorphic to the finite product $(-)^n$, for $n = |E|$. Therefore the power functor $(-)^{\underline{E}}$, preserves Cauchy completeness and can be restricted to an endofunctor on **CMet**. Thus also the reader monad restricts to **CMet**.

Because $\mathcal{R}$ is a continuous quantitative theory, the free monad on $\mathcal{R}$ in **CMet** is $\mathbb{C}T_{\mathcal{R}}$. Thus, by restricting Theorem 21 to quantitative algebras over **CMet**, we obtain:

▶ **Theorem 24.** *The monads* $\mathbb{C}T_{\mathcal{R}}$ *and* $(-)^{\underline{E}}$ *in* **CMet** *are isomorphic.*

In virtue of the above characterisation, by instantiating Theorem 22 in the category of complete extended metric spaces, in combination with Theorems 17 we obtain the following variant of the quantitative reader theory transformer on continuous quantitative theories.

▶ **Corollary 25.** *Let* $\mathcal{U}$ *be a continuous quantitative theory. Then,* $(\mathbb{C}T_{\mathcal{U}}-)^{\underline{E}}$ *is the free monad on the theory* $\mathcal{U} \otimes \mathcal{R}$ *in* **CMet**.

7 Quantitative Writer Algebras

Fix an extended metric space $\Lambda \in \mathbf{Met}$ of *output values* having monoid structure $(\Lambda, *, 0)$ with non-expansive multiplication operation $*\colon \Lambda \times \Lambda \to \Lambda$.

The *quantitative writer algebras* of type Λ are the algebras for the signature

$$\Sigma_{\mathcal{W}_\Lambda} = \{\mathsf{w}_\alpha \colon 1 \mid \alpha \in \Lambda\}$$

having a unary operator w_α, for each output value $\alpha \in \Lambda$, and satisfying the following axioms

$$\begin{aligned}
&(\mathsf{Zero}) \ \vdash x \equiv_0 \mathsf{w}_0(x)\,,\\
&(\mathsf{Mult}) \ \vdash \mathsf{w}_\alpha(\mathsf{w}_{\alpha'}(x)) \equiv_0 \mathsf{w}_{\alpha * \alpha'}(x)\,,\\
&(\mathsf{Diff}) \ \{x \equiv_\varepsilon x'\} \vdash \mathsf{w}_\alpha(x) \equiv_\delta \mathsf{w}_{\alpha'}(x')\,, \ \text{for } \delta \geq d_\Lambda(\alpha, \alpha') + \varepsilon\,.
\end{aligned}$$

The quantitative theory induced by the axioms above, written $\mathcal{W}_\Lambda$, is called *quantitative theory of writing computations*.

The term $\mathsf{w}_\alpha(t)$ represents the computation that proceeds as t after writing α on the output tape. The axiom (Zero) says that writing the identity element 0 is not observable on the tape; (Mult) says that consecutive writing operations are stored in the tape in the order of execution; (Diff) compares two computations w.r.t. the distance of their output values.

In the following, when the metric space Λ of output values is clear from the context, we use $\mathcal{W}$ in place of $\mathcal{W}_\Lambda$.

On Metric Spaces

Let $(\Lambda \,\square\, -)\colon \mathbf{Met} \to \mathbf{Met}$ be the functor assigning to each extended metric space X the space $(\Lambda \,\square\, X)$. By exploiting the monoid structure of Λ, the functor $(\Lambda \,\square\, -)$ can be given a monad structure with unit $\tau\colon Id \Rightarrow (\Lambda \,\square\, -)$ and multiplication $\varsigma\colon (\Lambda \,\square\, (\Lambda \,\square\, -)) \Rightarrow (\Lambda \,\square\, -)$, respectively given as follows, for arbitrary $x \in X$ and $\alpha, \alpha' \in \Lambda$

$$\tau_X(x) = (0, x)\,, \qquad\qquad \varsigma_X((\alpha, (\alpha', x))) = (\alpha * \alpha', x)\,.$$

This monad is also known as *writer monad* (also called *complexity monad*). Note that, the non-expansiveness of the maps above crucially depends on the assumption that the multiplication $*$ in Λ is non-expansive.

The next theorem says that the writer monad $(\Lambda \,\square\, -)$ has a quantitative equational presentation in terms of the theory $\mathcal{W}$ of writing computations.

▶ **Theorem 26.** *The monads* $T_{\mathcal{W}}$ *and* $(\Lambda \,\square\, -)$ *in* **Met** *are isomorphic.*

Let T be a strong monad with strength t. There is a canonical distributive law of the monad $(\Lambda \square -)$ over T, obtained using the strength $t_{\Lambda,-} : (\Lambda \square T-) \Rightarrow T(\Lambda \square -)$ of T. So $T(\Lambda \square -)$ acquires a canonical monad structure [2], and we obtain Moggi's writer monad transformer $T \mapsto T(\Lambda \square -)$ in **Met**.

In [12], Hyland et al. observed that Moggi's writer monad transformer can be equivalently recovered as the operation of tensoring with the writer monad.

▶ **Theorem 27** (Tensoring with Writer Monad [12]). *Let T be a strong monad. Then, the monad composition $T(\Lambda \square -)$ is given as $T \otimes (\Lambda \square -)$.*

By combining the above with Theorems 17 and 26, we get an analogous transformer at the level of quantitative equational theories as follows:

▶ **Corollary 28.** *Let $\mathcal{U}$ be a basic quantitative theory. Then, $T_{\mathcal{U}}(\Lambda \square -)$ is the free monad on the theory $\mathcal{U} \otimes \mathcal{W}$ in* **Met**.

On Complete Metric Spaces

If we assume the monoid $(\Lambda, *, 0)$ to be over a complete extended metric space Λ, the writer monad $(\Lambda \square -)$ is well defined also in **CMet**.

Since $\mathcal{W}$ is axiomatised by a continuous schema of quantitative conditional equations the free monad on $\mathcal{W}$ in **CMet** is given by $\mathbb{C}T_{\mathcal{W}}$. Thus, by restricting the use of Theorem 26 to quantitative algebras over complete extended metric spaces, we obtain:

▶ **Theorem 29.** *The monads $\mathbb{C}T_{\mathcal{W}}$ and $(\Lambda \square -)$ in* **CMet** *are isomorphic.*

Thus, by similar arguments as before, we obtain the following variant of Corollary 28.

▶ **Corollary 30.** *Let $\mathcal{U}$ be a continuous quantitative theory. Then, $\mathbb{C}T_{\mathcal{U}}(\Lambda \square -)$ is the free monad on the theory $\mathcal{U} \otimes \mathcal{W}$ in* **CMet**.

8 The Algebras of Labeled Markov Processes

In this section we show how to obtain a quantitative equational axiomatization of labelled Markov processes with discounted bisimilarity metric as the combination, via sum and tensor, of the following simpler quantitative equational theories:

(a) *The quantitative theory of interpolative barycentric algebras $\mathcal{B}$* from [21] over the signature $\Sigma_{\mathcal{B}} = \{+_e : 2 \mid e \in [0,1]\}$ extends M. H. Stone's theory of barycentric algebras [31] (a.k.a. abstract convex algebras) with the following axiom

$$\text{(IB)} \quad \{x \equiv_\varepsilon y, x' \equiv_{\varepsilon'} y'\} \vdash x +_e x' \equiv_\delta y +_e y', \quad \text{for } \delta \geq e\varepsilon + (1-e)\varepsilon'$$

expressing that the distance between convex combinations is obtained as the convex interpolation of the distance of their sub-terms. This theory will be used to axiomatise probability distributions with Kantorovich metric [15].

(b) *The pointed quantitative theory*, defined as the free quantitative theory $\mathcal{U}_{\mathbf{0}}$ (*i.e.*, the one imposing no additional axioms) for a signature $\Sigma_{\mathbf{0}} = \{\mathbf{0} : 0\}$ consisting of a single constant $\mathbf{0}$ symbol. This will be used to axiomatise termination.

(c) *The quantitative theory $\mathcal{R}_A$ of reading computations* (*cf.* Section 6) will be used to axiomatise the reaction to the choice of a label from a set A of action labels.

(d) *The quantitative theory of contractive operators* discussed in [1], is the theory obtained by imposing a Lipschitz contractive axiom for each operator in the signature. In our case, we consider a signature $\Sigma_\diamond = \{\diamond\colon 1\}$ with only one unary operator and the contractive theory $\mathcal{U}_\diamond$ generated from the axiom

$$(\diamond\text{-Lip}) \quad \{x =_\varepsilon y\} \vdash \diamond(x) \equiv_\delta \diamond(y)\,, \text{ for } \delta \geq c\varepsilon\,,$$

where $c \in (0,1)$ is a fixed *contractive factor* for the operator $\diamond$. This theory will be used to axiomatise the transition to a next state with discount factor c.

Formally, we define the quantitative theory $\mathcal{U}_{\mathbf{LMP}}$ of labelled Markov processes as the following combination of quantitative theories, with signature $\Sigma_{\mathbf{LMP}}$ given by the disjoint union of those from its component theories:

$$\Sigma_{\mathbf{LMP}} = \Sigma_{\mathcal{B}} + \Sigma_{\mathbf{0}} + \Sigma_{\mathcal{R}_A} + \Sigma_\diamond\,, \qquad \mathcal{U}_{\mathbf{LMP}} = ((\mathcal{B} + \mathcal{U}_{\mathbf{0}}) \otimes \mathcal{R}_A) + \mathcal{U}_\diamond\,.$$

Following [32, Section 6], we regard A-labelled Markov processes over extended metric spaces as $(\Delta(1+-))^{\underline{A}}$-coalgebras in **Met**, where Δ is the *Kantorovich functor* assigning to each $X \in \mathbf{Met}$ the space of Radon probability measures with finite moment over X equipped with Kantorovich metric. In [32] it is shown that the *probabilistic bisimilarity distance* on a labelled Markov processes (X,τ) is equal to the (pseudo)metric

$$\mathbf{d}_{(X,\tau)}(x,x') = d_Z(h(x),h(x'))\,,$$

where $h\colon X \to Z$ is the unique homomorphism to the final coalgebra (Z,ω).

Similarly to [1], we slightly extend the type of the coalgebras to encompass the case when the probabilistic bisimilarity distance is discounted by a factor $0 < c < 1$. Explicitly, we consider coalgebras for the functor $(\Delta(1 + c\cdot -))^{\underline{A}}$, where $(c\cdot -)$ is the *rescaling functor*, mapping a metric space (X,d_X) to $(X, c\cdot d_X)$. This will not change the essence of the results from [32] that are used in this section to characterise the probabilistic bisimilarity metric.

In the reminder of the section we prove that the theory $\mathcal{U}_{\mathbf{LMP}}$ axiomatizes (the monad of) A-labelled Markov processes with c-discounted bisimilarity metric.

On Metric Spaces

We characterise the monad $T_{\mathcal{U}_{\mathbf{LMP}}}$ in steps. First, note that $T_{\mathcal{U}_{\mathbf{0}}} \cong 1^* = (1+-)$ is the *maybe monad*, *i.e.*, freely generated monad on the constant terminal object functor 1. As the monad $(1+-)$ is isomorphic to $(1F)^*$, for any functor F, by [1, Theorems 4.4 and 5.2], and [12, Theorem 4], we obtain the following isomorphism of monads in **Met**:

$$T_{\mathcal{B}+\mathcal{U}_{\mathbf{0}}} \cong T_{\mathcal{B}} + T_{\mathcal{U}_{\mathbf{0}}} \cong \Pi(1+-)\,,$$

where $\Pi(1+-)$ is the *finite sub-distribution monad* with functor assigning to $X \in \mathbf{Met}$ the space of finitely supported Borel sub-probability measures with Kantorovich metric. Thus, $\mathcal{B} + \mathcal{U}_{\mathbf{0}}$ axiomatizes finitely supported sub-probability distributions with Kantorovich metric.

From the above, Theorem 17 and Corollary 23, we further get the monad isomorphism

$$T_{(\mathcal{B}+\mathcal{U}_{\mathbf{0}})\otimes\mathcal{R}_A} \cong \Pi(1+-) \otimes (-)^{\underline{A}} \cong (\Pi(1+-))^{\underline{A}}\,,$$

saying that tensoring with the theory $\mathcal{R}_A$ of reading computations corresponds to axiomatically adding the capability of reacting to the choice of an action label.

By [1, Theorem 6.3], $T_{\mathcal{U}_\diamond}$ is isomorphic to the free monad over the rescaling functor $(c\cdot -)$. Hence, by [1, Theorem 4.4] and [12, Corollary 2] we get the following last isomorphism

$$T_{\mathcal{U}_{\mathbf{LMP}}} = T_{((\mathcal{B}+\mathcal{U}_{\mathbf{0}})\otimes\mathcal{R}_A)+\mathcal{U}_\diamond} \cong \mu y.(\Pi(1 + c \cdot y + -))^{\underline{A}}.$$

Explicitly, this means that, the free monad on $\mathcal{U}_{\mathbf{LMP}}$ assigns to an arbitrary metric space $X \in \mathbf{Met}$ the *initial solution* of the following functorial equation in $\mathbf{Met}$

$$LMP_X \cong (\Pi(1 + c \cdot LMP_X + X))^{\underline{A}}.$$

In particular, when $X = 0$ is the empty metric space (*i.e.*, the initial object in $\mathbf{Met}$) the above corresponds to the isomorphism on the initial $(\Pi(1+c\cdot -))^{\underline{A}}$-algebra. The isomorphism gives us also a $(\Pi(1 + c \cdot -))^{\underline{A}}$-coalgebra structure on LMP_0, which can be converted into a labeled Markov process (LMP_0, τ_0) via a post-composition with the inclusion $\Pi(-) \hookrightarrow \Delta(-)$.

The key aspect is that the metric of LMP_0 is exactly the bisimilarity metric.

▶ **Lemma 31.** *d_{LMP_0} is the c-discounted probabilistic bisimilarity metric on (LMP_0, τ_0).*

▶ Remark 32. For a less abstract description of (LMP_0, τ_0), notice that the elements of LMP_0 are (equivalence classes of) ground terms over the signature $\Sigma_{\mathbf{LMP}}$, which one can interpret as pointed (or rooted) acyclic labelled Markov processes quotiented by bisimilarity.

On Complete Metric Spaces

Since all the quantitative theories considered are continuous, we can replicate the same steps also while interpreting the theory $\mathcal{U}_{\mathbf{LMP}}$ over complete metric spaces, obtaining the monad

$$\mathbb{C}T_{\mathcal{U}_{\mathbf{LMP}}} \cong \mu y.\Delta(1 + c \cdot y + -)^{\underline{A}}.$$

By following similar arguments to [1, Section 8.3], one can prove that the functorial equation $LMP_X \cong \Delta(1 + c \cdot LMP_X + X)^{\underline{A}}$ has a unique solution. Thus by applying the monad above on $X = 0$ we recover the carrier of the final $(\Delta(1+c\cdot -))^{\underline{A}}$-coalgebra, equipped with c-discounted probabilistic bisimilarity metric.

▶ Remark 33. While by interpreting the theory $\mathcal{U}_{\mathbf{LMP}}$ over $\mathbf{Met}$ we can only characterise Markov processes that are acyclic, by doing it over $\mathbf{CMet}$ we obtain an algebraic representation of all bisimilarity classes as the elements of the final coalgebra. Thus, among others, we also recover Markov processes with cyclic structures as the limit of all their finite unfoldings.

9 The Algebras of Markov Decision Processes with Rewards

As a last example, we provide a quantitative axiomatization of Markov decision processes with rewards equipped with discounted bisimilarity metric. As the construction is similar to Section 8, we avoid repeating the details of each step of the monad characterization.

Let $(\mathbb{R}, +, 0)$ be the standard monoid structure on the reals. We define the quantitative theory $\mathcal{U}_{\mathbf{MDP}}$ of Markov decision processes with real-valued rewards as follows

$$\Sigma_{\mathbf{MDP}} = \Sigma_{\mathcal{B}} + \Sigma_{\mathcal{W}_{\mathbb{R}}} + \Sigma_{\mathcal{R}_A} + \Sigma_\diamond\,, \qquad \mathcal{U}_{\mathbf{MDP}} = ((\mathcal{B} \otimes \mathcal{U}_{\mathcal{W}_{\mathbb{R}}}) \otimes \mathcal{R}_A) + \mathcal{U}_\diamond\,,$$

where $\mathcal{W}_{\mathbb{R}}$ is the theory of writing computations and the other theories are as in Section 8.

For convenience, we regard Markov decision processes over metric spaces as the coalgebras for the functor $(\Delta(\mathbb{R} \,\square\, c \cdot -))^{\underline{A}}$ on $\mathbf{Met}$, where the endofunctor $(\mathbb{R} \,\square\, -)$ is used to encode the metric differences at each decision step for the real-valued reward available for two states. Via this coalgebraic representation, the c-discounted *probabilistic bisimilarity distance* on this structures can be defined as in [32] (following the same definition of Section 8).

▶ Remark 34. In [30] a Markov decision process is defined as a tuple $(S, p(\cdot|s,a), r(s,a))$ with a Markov kernel $p\colon S \times A \to \Delta(S)$ and randomised reward function $r\colon S \times A \to \Delta(\mathbb{R})$. Our coalgebraic representation is the natural generalisation over metric spaces, where the randomness of the Markov kernel and reward function is combined as a probability measure on $(\mathbb{R} \,\square\, c \cdot S)$, by regarding $\mathbb{R}$ and S as extended metric spaces (for each $a \in A$).

On Metric Spaces and Complete Metric Spaces

Similarly to what we have done in Section 8 for labelled Markov processes, we relate Markov decision processes and their c-discounted probabilistic bisimilarity pseudometric with the free monads on the theory $\mathcal{U}_{\mathbf{MDP}}$ in **Met** and **CMet**.

The only step that changes in the characterisation of $T_{\mathcal{U}_{\mathbf{MDP}}}$, regards the combination of theories $\mathcal{B} \otimes \mathcal{U}_{\mathcal{W}_{\mathbb{R}}}$, which is dealt using Corollary 28. Thus, similarly to Section 8 we get

$$T_{\mathcal{U}_{\mathbf{MDP}}} = T_{((\mathcal{B}+\otimes\mathcal{U}_{\mathcal{W}_{\mathbb{R}}})\otimes\mathcal{R}_A)+\mathcal{U}_\diamond} \cong \mu y.\Pi((\mathbb{R} \,\square\, y) + -)^{\underline{A}}.$$

The metric on the initial solution for the functorial fixed point definition corresponds to the c-discounted probabilistic bisimilarity (pseudo)metric on its coalgebra structure.

Similar considerations apply also when interpreting the theories in the category **CMet** of complete metric spaces, as the argument follows without issues because $\mathbb{R}$ a complete metric space. Thus we obtain the following characterisation for the monad:

$$\mathbb{C}T_{\mathcal{U}_{\mathbf{LMP}}} \cong \mu y.\Delta((\mathbb{R} \,\square\, y) + -)^{\underline{A}}.$$

Again, the metric on the solution for the above functorial fixed point definition corresponds to the c-discounted probabilistic bisimilarity metric. Moreover, as the fixed point solution is unique, $\mathbb{C}T_{\mathcal{U}_{\mathbf{LMP}}}0$ is an algebraic characterization of the final $(\Delta(\mathbb{R} \,\square\, c \cdot -))^{\underline{A}}$-coalgebra.

10 Conclusions

We studied the commutative combination of quantitative effects as the tensor of their quantitative equational theories. The key result in this regard is Theorem 17, asserting that the tensor of two quantitative theories corresponds to the categorical tensor of their free monads. In addition to this general result, we show how to extend to the quantitative algebraic setting Moggi's notions of reader and writer monad transformers.

We illustrate the applicability of our theoretical development with two examples: labeled Markov processes and Markov decision processes. Apart from the intrinsic interest in their quantitative equational axiomatisations, what is particularly pleasant is the systematic compositional way with which one can obtain quantitative axiomatisations of different variants of Markov processes by just combining theories as new basic ingredients.

An example that escapes our compositional treatment via sum and tensor is the combination of probabilities and non-determinism as illustrated in [23]. A possible future work in this direction is to extend the combination of theories with another operator: the distributive tensor (see [13, Section 6]). Following a similar intuition by Cheng [6], we claim that these correspond in a suitable way to Garner's weak distributive law [9]. Our claim seems promising in the light of the work [10, 3] which consider equational axiomatisations combining probabilities and non-determinism.

References

1 Giorgio Bacci, Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. An algebraic theory of markov processes. In *LICS*, pages 679–688. ACM, 2018.

2 Jon Beck. Distributive laws. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of Lect. Notes Math., pages 119–140. Springer, 1966.

3 Filippo Bonchi and Alessio Santamaria. Combining semilattices and semimodules. *CoRR*, abs/2012.14778, 2020.

4 Nathan J. Bowler, Sergey Goncharov, Paul Blain Levy, and Lutz Schr "o der. Exploring the boundaries of monad tensorability on set. *Log. Methods Comput. Sci.*, 9(3), 2013.

5 Pietro Cenciarelli and Eugenio Moggi. A syntactic approach to modularity in denotational semantics. Technical report, CWI, 1993. Proc. 5th. Biennial Meeting on Category Theory and Computer Science.

6 Eugenia Cheng. Distributive laws for lawvere theories. *Compositionality*, 2:1, May 2020. doi:10.32408/compositionality-2-1.

7 Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. Metrics for labelled Markov processes. *Theoretical Computer Science*, 318(3):323–354, 2004.

8 Peter J. Freyd. Algebra valued functors in general and tensor products in particular. *Colloq. Math.*, 14:89–106, 1966.

9 Richard Garner. The vietoris monad and weak distributive laws. *Appl. Categorical Struct.*, 28(2):339–354, 2020.

10 Alexandre Goy and Daniela Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In *LICS*, pages 454–464. ACM, 2020.

11 Martin Hyland, Paul Blain Levy, Gordon D. Plotkin, and John Power. Combining algebraic effects with continuations. *Theor. Comput. Sci.*, 375(1-3):20–40, 2007.

12 Martin Hyland, Gordon D. Plotkin, and John Power. Combining effects: Sum and tensor. *Theor. Comput. Sci.*, 357(1-3):70–99, 2006. doi:10.1016/j.tcs.2006.03.013.

13 Martin Hyland and John Power. Discrete lawvere theories and computational effects. *Theor. Comput. Sci.*, 366(1-2):144–162, 2006.

14 Martin Hyland and John Power. The category theoretic understanding of universal algebra: Lawvere theories and monads. *Electronic Notes in Theor. Comp. Sci.*, 172:437–458, 2007.

15 Leonid Vitalevich Kantorovich. On the transfer of masses (in Russian). *Doklady Akademii Nauk*, 5(5-6):1–4, 1942. Translated in Management Science, 1958.

16 Gregory M. Kelly. Basic concepts of enriched category theory. *Theory and Applications of Categories*, 1982. Reprinted in 2005.

17 Anders Kock. Strong functors and monoidal monads. *Arch. Math. (Basel)*, 23:113–120, 1972.

18 Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 2nd edition, 1998.

19 William F. Lawvere. Metric spaces, generalized logic, and closed categories. In *Seminario Mat. e. Fis. di Milano*, volume 43, pages 135–166. Springer, 1973.

20 Ernest Manes. A Triple Theoretic Construction of Compact Algebras. In *Seminar on Triples and Categorical Homology Theory*, volume 80 of Lect. Notes Math., pages 91–118. Springer, 1966.

21 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. Quantitative Algebraic Reasoning. In *LICS*, pages 700–709. ACM, 2016. doi:10.1145/2933575.2934518.

22 Radu Mardare, Prakash Panangaden, and Gordon D. Plotkin. On the axiomatizability of quantitative algebras. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005102.

23 Matteo Mio and Valeria Vignudelli. Monads and quantitative equational theories for nondeterminism and probability. In *CONCUR*, volume 171 of *LIPIcs*, pages 28:1–28:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

24 Eugenio Moggi. *The partial lambda calculus*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics., 1988.

25 Eugenio Moggi. Notions of computation and monads. *Information and computation*, 93(1):55–92, 1991.

26 Gordon Plotkin and John Power. Semantics for algebraic operations. *Electronic Notes in Theoretical Computer Science*, 45:332–345, 2001.

27 Gordon Plotkin and John Power. Notions of computation determine monads. In *Foundations of Software Science and Computation Structures*, pages 342–356. Springer, 2002.

28 Gordon D. Plotkin and John Power. Semantics for algebraic operations. In *MFPS*, volume 45 of *Electronic Notes in Theoretical Computer Science*, pages 332–345. Elsevier, 2001.

29 Gordon D. Plotkin and John Power. Algebraic operations and generic effects. *Appl. Categorical Struct.*, 11(1):69–94, 2003.

30 M. L. Puterman. *Markov Decision Processes*. Wiley, 2005.

31 Marshall H. Stone. Postulates for the barycentric calculus. *Annali di Matematica Pura ed Applicata*, 29(1):25–30, 1949.

32 Franck van Breugel, Claudio Hermida, Michael Makkai, and James Worrell. Recursively defined metric spaces without contraction. *Theor. Comput. Sci.*, 380(1-2):143–163, 2007. doi:10.1016/j.tcs.2007.02.059.

Pushdown Automata and Context-Free Grammars in Bisimulation Semantics

Jos C. M. Baeten
CWI, Amsterdam, The Netherlands
University of Amsterdam, The Netherlands

Cesare Carissimo
University of Amsterdam, The Netherlands

Bas Luttik
Eindhoven University of Technology, The Netherlands

Abstract

The Turing machine models an old-fashioned computer, that does not interact with the user or with other computers, and only does batch processing. Therefore, we came up with a Reactive Turing Machine that does not have these shortcomings. In the Reactive Turing Machine, transitions have labels to give a notion of interactivity. In the resulting process graph, we use bisimilarity instead of language equivalence.

Subsequently, we considered other classical theorems and notions from automata theory and formal languages theory. In this paper, we consider the classical theorem of the correspondence between pushdown automata and context-free grammars. By changing the process operator of sequential composition to a sequencing operator with intermediate acceptance, we get a better correspondence in our setting. We find that the missing ingredient to recover the full correspondence is the addition of a notion of state awareness.

2012 ACM Subject Classification Theory of computation → Interactive computation; Theory of computation → Turing machines; Theory of computation → Grammars and context-free languages; Theory of computation → Process calculi

Keywords and phrases pushdown automaton, context-free grammar, bisimilarity, intermediate acceptance, state awareness

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.8

1 Introduction

A basic ingredient of any undergraduate curriculum in computer science is a course on automata theory and formal languages, as this gives students insight in the essence of a computer, and tells them what a computer can and cannot do. Usually, such a course contains the treatment of the Turing machine as an abstract model of a computer. However, the Turing machine is a very old-fashioned computer: it is deaf, dumb and blind, and all input from the user has to be put on the tape before the start. Computers behaved like this until the advent of the terminal in the mid 1970s. This is far removed from computers the students find all around them, which interact continuously with people, other computers and the internet. It is hard to imagine a self-driving car driven by a Turing machine that is deaf, dumb and blind, where all user input must be on the tape at the start of the trip.

In order to make the Turing machine more interactive, many authors have enhanced it with extra features, see e.g. [10, 16]. But an extra feature, we believe, is not the way to go. Interaction is an essential ingredient, such as it has been treated in many forms of concurrency theory. We seek a full integration of automata theory and concurrency theory, and proposed the Reactive Turing Machine in [4]. In the Reactive Turing Machine, transitions have labels to give a notion of interactivity. In the resulting process graphs, we use bisimilarity instead of language equivalence.

Subsequently, we considered other classical theorems and notions from automata theory and formal languages theory [2]. We find richer results and a finer theory. In this paper, we consider the classical theorem of the correspondence between pushdown automata and context-free grammars. Before [3], we did not get a good correspondence in the process setting. By changing the process operator of sequential composition to a sequencing operator with intermediate acceptance, we get a better correspondence in our setting [5, 7, 8]. We find that the missing ingredient to recover the full correspondence is the addition of a notion of state awareness, by means of a signal that can be passed along a sequencing operator.

2 Preliminaries

As a common semantic framework we use the notion of a *labelled transition system*.

▶ **Definition 1.** *A* labelled transition system *is a quadruple* $(\mathcal{S}, \mathcal{A}, \rightarrow, \downarrow)$*, where*
1. $\mathcal{S}$ *is a set of* states*;*
2. $\mathcal{A}$ *is a set of* actions*,* $\tau \notin \mathcal{A}$*;*
3. $\rightarrow \subseteq \mathcal{S} \times \mathcal{A} \cup \{\tau\} \times \mathcal{S}$ *is an* $\mathcal{A} \cup \{\tau\}$*-labelled* transition relation*; and*
4. $\downarrow \subseteq \mathcal{S}$ *is the set of final or accepting states.*

A process graph *is a labelled transition system with a special designated* root state $\uparrow$*, i.e., it is a quintuple* $(\mathcal{S}, \mathcal{A}, \rightarrow, \uparrow, \downarrow)$ *such that* $(\mathcal{S}, \mathcal{A}, \rightarrow, \downarrow)$ *is a labelled transition system, and* $\uparrow \in \mathcal{S}$*. We write* $s \xrightarrow{a} s'$ *for* $(s, a, s') \in \rightarrow$ *and* $s\downarrow$ *for* $s \in \downarrow$*.*

By considering language equivalence classes of process graphs, we recover languages as a semantics, but we can also consider other equivalence relations. Notable among these is *bisimilarity*.

▶ **Definition 2.** *Let* $(\mathcal{S}, \mathcal{A}, \rightarrow, \downarrow)$ *be a labelled transition system. A symmetric binary relation* R *on* $\mathcal{S}$ *is a* bisimulation *if it satisfies the following conditions for every* $s, t \in \mathcal{S}$ *such that* $s \mathrel{R} t$ *and for all* $a \in \mathcal{A} \cup \{\tau\}$*:*
1. *if* $s \xrightarrow{a} s'$ *for some* $s' \in \mathcal{S}$*, then there is a* $t' \in \mathcal{S}$ *such that* $t \xrightarrow{a} t'$ *and* $s' \mathrel{R} t'$*; and*
2. *if* $s\downarrow$*, then* $t\downarrow$*.*

The results of this paper do not rely on abstraction from internal computations, so we can use the *strong* version of bisimilarity defined above, which does not give special treatment to τ-labelled transitions. But in general we have to use a version of bisimilarity that accomodates for abstraction from internal activity; the finest such notion of bisimilarity is *divergence-preserving branching bisimilarity*, which was introduced in [14] (see also [12] for an overview of recent results).

A *process* is a bisimulation equivalence class of process graphs.

3 Pushdown Automata

We consider an abstract model of a computer with a memory in the form of a *stack*: this stack can be accessed only at the top: something can be added on top of the stack (push), or something can be removed from the top of the stack (pop).

▶ **Definition 3** (pushdown automaton)**.** *A* pushdown automaton *M is a sextuple* $(\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ *where:*
1. $\mathcal{S}$ *is a finite set of states,*
2. $\mathcal{A}$ *is a finite input alphabet,* $\tau \notin \mathcal{A}$ *is the unobservable step,*
3. $\mathcal{D}$ *is a finite data alphabet,*

4. $\rightarrow \subseteq \mathcal{S} \times (\mathcal{A} \cup \{\tau\}) \times (\mathcal{D} \cup \{\epsilon\}) \times \mathcal{D}^* \times \mathcal{S}$ *is a finite set of* transitions *or* steps,
5. $\uparrow \in \mathcal{S}$ *is the initial state,*
6. $\downarrow \subseteq \mathcal{S}$ *is the set of final or accepting states.*

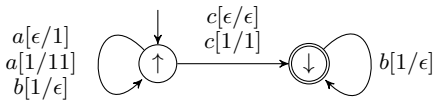


Figure 1 An example pushdown automaton.

If $(s, a, d, x, t) \in \rightarrow$ with $d \in \mathcal{D}$, we write $s \xrightarrow{a[d/x]} t$, and this means that the machine, when it is in state s and d is the top element of the stack, can consume input symbol a, replace d by the string x and thereby move to state t. Likewise, writing $s \xrightarrow{a[\epsilon/x]} t$ means that the machine, when it is in state s and the stack is empty, can consume input symbol a, put the string x on the stack and thereby move to state t. In steps $s \xrightarrow{\tau[d/x]} t$ and $s \xrightarrow{\tau[\epsilon/x]} t$, no input symbol is consumed, only the stack is modified.

For example, consider the pushdown automaton depicted in Figure 1. It represents the process that can start to read an a or a c, and after it has read at least one a, can also read b's. Upon acceptance, it will have read just one c, up to as many b's as it has read a's, and no a's after reading the c.

We do not consider the language of a pushdown automaton, but rather consider the process, i.e., the bisimulation equivalence class of the process graph of a pushdown automaton. A state of this process graph is a pair (s, x), where $s \in \mathcal{S}$ is the current state and $x \in \mathcal{D}^*$ is the current contents of the stack (the left-most element of x being the top of the stack). In the initial state, the stack is empty. In a final state, acceptance can take place irrespective of the contents of the stack. The transitions in the process graph are labeled by the inputs of the pushdown automaton or τ.

▶ **Definition 4.** *Let* $M = (\mathcal{S}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ *be a pushdown automaton. The* process graph $\mathcal{P}(M) = (\mathcal{S}_{\mathcal{P}(M)}, \mathcal{A}, \rightarrow_{\mathcal{P}(M)}, \uparrow_{\mathcal{P}(M)}, \downarrow_{\mathcal{P}(M)})$ *associated with* M *is defined as follows:*

1. $\mathcal{S}_{\mathcal{P}(M)} = \{(s, x) \mid s \in \mathcal{S} \;\&\; x \in \mathcal{D}^*\}$;
2. $\rightarrow_{\mathcal{P}(M)} \subseteq \mathcal{S}_{\mathcal{P}(M)} \times \mathcal{A} \cup \{\tau\} \times \mathcal{S}_{\mathcal{P}(M)}$ *is the least relation such that for all* $s, s' \in \mathcal{S}$, $a \in \mathcal{A} \cup \{\tau\}$, $d \in \mathcal{D}$ *and* $x, x' \in \mathcal{D}^*$ *we have*

$$(s, dx) \xrightarrow{a}_{\mathcal{P}(M)} (s', x'x) \text{ if, and only if, } s \xrightarrow{a[d/x']} s' \;;$$

$$(s, \epsilon) \xrightarrow{a}_{\mathcal{P}(M)} (s', x) \text{ if, and only if, } s \xrightarrow{a[\epsilon/x]} s' \;;$$

3. $\uparrow_{\mathcal{P}(M)} = (\uparrow, \epsilon)$;
4. $\downarrow_{\mathcal{P}(M)} = \{(s, x) \mid s \in \downarrow \;\&\; x \in \mathcal{D}^*\}$.

To distinguish, in the definition above, the set of states, the transition relation, the initial state and the set of accepting states of the pushdown automaton from similar components of the associated process graph, we have attached a subscript $\mathcal{P}(M)$ to the latter. In the remainder of this paper, we will suppress the subscript whenever it is already clear from the context whether a component of the pushdown automaton or its associated transition system is meant.

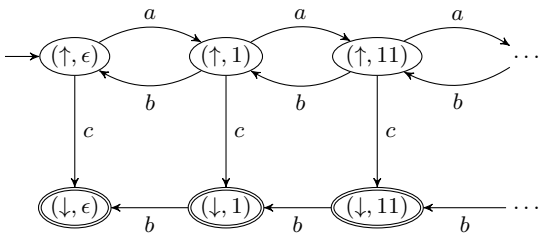


Figure 2 The process graph associated with the pushdown automaton in Figure 1.

Figure 2 depicts the process graph associated with the pushdown automaton depicted in Figure 1.

By adding additional states and τ-transitions, it is enough to consider only *push* and *pop* transitions: a push transition is of the form $s \xrightarrow{a[d/ed]} t$ or $s \xrightarrow{a[\epsilon/d]} t$, where one data element is added on top of the stack, and a pop transition is of the form $s \xrightarrow{a[d/\epsilon]} t$, where the top of the stack is removed ($a \in (\mathcal{A} \cup \{\tau\})$), see [2].

4 Sequential Processes

In our setting, a context-free grammar is denoted by a finite guarded recursive specification over the process theory of sequential expressions.

In this section we present the Theory of Sequential Processes adopting the revised operational semantics for sequential composition proposed in [5]. Sequential composition with the operational semantics of [6] is denoted by $\cdot$, and we call the operator with the revised operational semantics *sequencing* and denote it by ;.

Let $\mathcal{A}$ be a set of *actions* and $\tau \notin \mathcal{A}$ *the silent action*, symbols denoting atomic events, and let $\mathcal{P}$ be a finite set of *process identifiers*. The sets $\mathcal{A}$ and $\mathcal{P}$ serve as parameters of the process theory that we shall introduce below. The set of *sequential process expressions* is generated by the following grammar ($a \in \mathcal{A} \cup \{\tau\}$, $X \in \mathcal{P}$):

$$p ::= \mathbf{0} \mid \mathbf{1} \mid a.p \mid p + p \mid p\,;p \mid X \quad .$$

The constants $\mathbf{0}$ and $\mathbf{1}$ respectively denote the *deadlocked* (i.e., inactive but not accepting) process and the *accepting* process. For each $a \in \mathcal{A} \cup \{\tau\}$ there is a unary action prefix operator $a._-$. The binary operators $+$ and ; denote alternative composition and sequencing, respectively. We adopt the convention that $a._-$ binds strongest and $+$ binds weakest. For a (possibly empty) sequence $p_1, \ldots, p_n$ we inductively define $\sum_{i=1}^{n} p_i = \mathbf{0}$ if $n = 0$ and $\sum_{i=1}^{n} p_i = (\sum_{i=1}^{n-1} p_i) + p_n$ if $n > 0$. The symbol ; is often omitted when writing process expressions. In particular, if $\alpha \in \mathcal{P}^*$, say $\alpha = X_1 \cdots X_n$, then α denotes the process expression inductively defined by $\alpha = \mathbf{1}$ if $n = 0$ and $\alpha = (X_1 \cdots X_{n-1})\,;X_n$ if $n > 0$.

A recursive specification over sequential process expressions is a mapping Δ from $\mathcal{P}$ to the set of sequential process expressions. The idea is that the process expression p associated with a process identifier $X \in \mathcal{P}$ by Δ *defines* the behaviour of X. We prefer to think of Δ as a collection of *defining equations* $X \stackrel{\text{def}}{=} p$, exactly one for every $X \in \mathcal{P}$. We shall, throughout the paper, presuppose a recursive specification Δ defining the process identifiers in $\mathcal{P}$, and we shall usually simply write $X \stackrel{\text{def}}{=} p$ for $\Delta(X) = p$. Note that, by our assumption that $\mathcal{P}$ is finite, Δ is finite too.

$$\frac{}{a.p \xrightarrow{a} p} \qquad \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$$

$$\frac{}{\mathbf{1}\downarrow} \qquad \frac{p\downarrow}{(p+q)\downarrow} \qquad \frac{q\downarrow}{(p+q)\downarrow}$$

$$\frac{p\downarrow \quad q\downarrow}{p\,;q\downarrow} \qquad \frac{p \xrightarrow{a} p'}{p\,;q \xrightarrow{a} p'\,;q} \qquad \frac{p\downarrow \quad p \not\xrightarrow{} \quad q \xrightarrow{a} q'}{p\,;q \xrightarrow{a} q'}$$

$$\frac{p \xrightarrow{a} p' \quad X \stackrel{\text{def}}{=} p}{X \xrightarrow{a} p'} \qquad \frac{p\downarrow \quad X \stackrel{\text{def}}{=} p}{X\downarrow}$$

Figure 3 Operational semantics for sequential process expressions.

We associate behaviour with process expressions by defining, on the set of process expressions, a unary acceptance predicate $\downarrow$ (written postfix) and, for every $a \in \mathcal{A} \cup \{\tau\}$, a binary transition relation $\xrightarrow{a}$ (written infix), by means of the transition system specification presented in Fig. 3. We write $p \not\xrightarrow{a}$ for "there does not exist p' such that $p \xrightarrow{a} p'$" and $p \not\xrightarrow{}$ for "$p \not\xrightarrow{a}$ for all $a \in \mathcal{A} \cup \{\tau\}$".

For $w \in \mathcal{A}^*$ we define $p \xrightarrow{w}\!\!\!\!\!\rightarrow p'$ inductively, for all process expressions p, p', p'';

- $p \xrightarrow{\epsilon}\!\!\!\!\!\rightarrow p$;
- if $p \xrightarrow{a} p'$ and $p' \xrightarrow{w}\!\!\!\!\!\rightarrow p''$, then $p \xrightarrow{aw}\!\!\!\!\!\rightarrow p''$ $(a \in \mathcal{A})$;
- if $p \xrightarrow{\tau} p'$ and $p' \xrightarrow{w}\!\!\!\!\!\rightarrow p''$, then $p \xrightarrow{w}\!\!\!\!\!\rightarrow p''$.

We see that τ-steps do not contribute to the string w. We write $p \longrightarrow p'$ for there exists $a \in \mathcal{A} \cup \{\tau\}$ such that $p \xrightarrow{a} p'$. Similarly, we write $p \longrightarrow\!\!\!\!\!\rightarrow p'$ for there exists $w \in \mathcal{A}^*$ such that $p \xrightarrow{w}\!\!\!\!\!\rightarrow p'$ and say that p' is *reachable* from p.

It is well-known that transition system specifications with negative premises may not define a unique transition relation that agrees with provability from the transition system specification [11, 9, 13]. Indeed, in [5] it was already pointed out that the transition system specification in Fig. 3 gives rise to such anomalies, e.g., if Δ includes for X the defining equation $X \stackrel{\text{def}}{=} X\,;a.\mathbf{1} + \mathbf{1}$. For then, if $X \not\xrightarrow{}$, according to the rules for sequencing and recursion we find that $X \xrightarrow{a} \mathbf{1}$, which is a contradiction. On the other hand, the transition $X \xrightarrow{a} \mathbf{1}$ is not provable from the transition system specification.

We remedy the situation by restricting our attention to *guarded* recursive specifications, i.e., we require that every occurrence of a process identifier in the definition of some (possibly different) process identifier occurs within the scope of an action prefix. Note that we allow τ as a guard. This is possible since we use strong bisimulation, not branching bisimulation. If Δ is guarded, then it is straightforward to prove that the mapping S from process expressions to natural numbers inductively defined by $S(\mathbf{1}) = S(\mathbf{0}) = S(a.p) = 0$, $S(p_1 + p_2) = S(p_1\,;p_2) = S(p_1) + S(p_2) + 1$, and $S(X) = S(p)$ if $(X \stackrel{\text{def}}{=} p) \in \Delta$ gives rise to a so-called *stratification* S' from transitions to natural numbers defined by $S'(p \xrightarrow{a} p') = S(p)$ for all $a \in \mathcal{A} \cup \{\tau\}$ and process expressions p and p'. In [11] it is proved that whenever such a stratification exists, then the transition system specification defines a unique transition relation that agrees with provability in the transition system specification.

The operational rules in Fig. 3 deviate from the operational rules for the Theory of Sequential Processes discussed in [6] in only two ways: to get that set of rules, the symbol ; should be replaced by $\cdot$, and the negative premise $p \not\xrightarrow{}$ should be removed from the third sequencing rule. The replacement of ; by $\cdot$ is, of course, insignificant; the removal of the

negative premise $p \not\rightarrow$, however, does have a significant effect on the semantics of sequencing. The negative premise ensures that a sequencing can only proceed to execute its second argument when its first argument not only satisfies the acceptance predicate, but also cannot perform any further activity. The semantic difference between ; and $\cdot$ is illustrated in the following example.

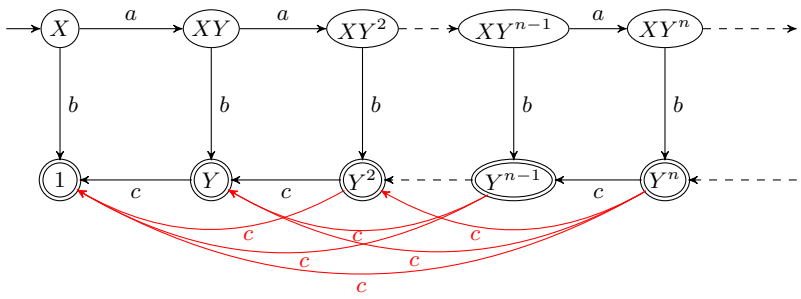


Figure 4 The difference between ; and $\cdot$.

▶ **Example 5.** Consider the recursive specification

$$X \stackrel{\text{def}}{=} a.(XY) + b.\mathbf{1} \qquad Y \stackrel{\text{def}}{=} c.\mathbf{1} + \mathbf{1} \ .$$

We have deliberately omitted the occurrence of the sequencing operator between X and Y from the right-hand side of the defining equation for X (as is, actually, standard practice). Depending on whether we interpret the sequencing of X and Y using the semantics for $\cdot$ or for ;, we obtain the process graph shown in Fig. 4 with or without the red c-transitions. Note that, under the $\cdot$-interpretation, the phenomenon of *transparency* plays a role: from Y^n we have c-transitions to every Y^k with $k < n$, by executing the c-transition of the kth occurrence of Y, thus skipping the first $k-1$ occurrences of Y. This behaviour is prohibited by the negative premise in the rule for ;, for, since $Y \stackrel{c}{\longrightarrow} \mathbf{1}$, none of the occurrences of Y can be skipped.

When a term p satisfies both $p\downarrow$ and $p \longrightarrow$ we say p has *intermediate acceptance*. We will need to take special care of such terms in the sequel.

We proceed to define when two closed terms are behaviourally equivalent.

▶ **Definition 6.** *A binary relation R on the set of sequential process expressions is a* bisimulation *iff R is symmetric and for all closed terms p and q such that if $(p,q) \in R$:*

1. *If $p \stackrel{a}{\longrightarrow} p'$, then there exists a term q', such that $q \stackrel{a}{\longrightarrow} q'$, and $(p',q') \in R$.*
2. *If $p\downarrow$, then $q\downarrow$.*

The terms p and q are bisimilar (notation: $p \leftrightarroweq q$) iff there exists a bisimulation R such that $(p,q) \in R$.

The operational rules presented in Fig 3 are in the so-called *panth format* from which it immediately follows that bisimilarity is a congruence [15].

▶ **Proposition 7.** *The relation $\leftrightarroweq$ is a congruence on sequential process expressions.*

5 The correspondence

The classical theorem states that a language can be defined by a push-down automaton just in case it can be defined by a context-free grammar. In our setting, we do have that the process of a given guarded sequential specification (i.e., the equivalence class of process graphs bisimilar to the process graph associated with the sequential specification) coincides with the process of some push-down automaton (i.e., the equivalence class of process graphs bisimilar to the process graph associated with the push-down automaton), but not the other way around: there is a push-down automaton of which the process is different from the process of any guarded sequential specification. In this section, we will prove these facts, in the next section, we investigate what is needed in addition to recover the full correspondence.

First of all, we look at the failing direction. It can fail if the push-down automaton has at least two states. For one state, it does work.

▶ **Theorem 8.** *For every one-state pushdown automaton there is a guarded sequential specification of which the process coincides with the process of the automaton.*

Proof. Let $M = (\{\uparrow\}, \mathcal{A}, \mathcal{D}, \rightarrow, \uparrow, \downarrow)$ be a pushdown automaton. We can assume $\rightarrow$ only has push and pop transitions. If there is no transition $\uparrow \xrightarrow{a[\epsilon/d]} \uparrow$, then we can take either $X = \mathbf{1}$ or $X = \mathbf{0}$ as the resulting specification (in case $\downarrow = \{\uparrow\}$ resp. $\downarrow = \emptyset$). Otherwise, add a summand $a.X_d \,;\, X$ for each such transition to the equation of the initial identifier X. Next, the equation for the added identifier X_d has a summand $a.\mathbf{1}$ for each transition $\uparrow \xrightarrow{a[d/\epsilon]} \uparrow$ and a summand $a.X_e \,;\, X_d$ for each transition $\uparrow \xrightarrow{a[d/ed]} \uparrow$, apart from a summand $\mathbf{1}$ or $\mathbf{0}$, depending on whether $\uparrow \in \downarrow$ or not. ◀

▶ **Example 9.** The stack that is accepting in every state has a pushdown automaton with state $\uparrow$, data some finite set D, actions $\{push_d, pop_d \mid d \in D\}$, $\downarrow = \{\uparrow\}$ and transitions $\uparrow \xrightarrow{push_d[\epsilon/d]} \uparrow$ and $\uparrow \xrightarrow{pop_d[d/\epsilon]} \uparrow$ for each $d \in D$ and transitions $\uparrow \xrightarrow{push_e[d/ed]} \uparrow$ for each $d, e \in D$.

The recursive specification becomes

$$X \stackrel{\text{def}}{=} \mathbf{1} + \sum_{d \in D} push_d.X_d \,;\, X \qquad X_d \stackrel{\text{def}}{=} \mathbf{1} + pop_d.\mathbf{1} + \sum_{e \in D} push_e.X_e \,;\, X_d \;\; (d \in D)$$

Note that if we use the sequential composition operator $\cdot$ of [6] instead of the present sequencing operator ;, then Theorem 8 fails because of the transparency illustrated in Figure 4. With the sequential composition operator, we cannot find a recursive specification of the stack accepting in every state of Example 9, because of the same phenomenon.

In order to prove that there is a push-down automaton, of which the process cannot be specified by a guarded sequential specification, it is convenient to present a guarded sequential specification in a normal form, the so-called Greibach normal form, see [8].

▶ **Definition 10.** *A guarded sequential specification is in Greibach normal form, GNF, if every right-hand side of every equation has one of the following two forms:*

- $\sum_{i=1}^{n} a_i.\alpha_i$ *for actions* $a_i \in \mathcal{A} \cup \{\tau\}$ *and identifiers* $\alpha_i \in \mathcal{P}^*$, $n \geq 0$.
- $\mathbf{1} + \sum_{i=1}^{n} a_i.\alpha_i$ *for actions* $a_i \in \mathcal{A} \cup \{\tau\}$ *and identifiers* $\alpha_i \in \mathcal{P}^*$, $n \geq 0$.

Recall that the empty summation equals $\mathbf{0}$ *and the empty sequence is* $\mathbf{1}$.

By adding a finite number of process identifiers, every guarded sequential specification can be brought into Greibach normal form (i.e. the behaviour associated with a process identifier by the original specification is strongly bisimilar to the behaviour associated with it by the transformed specification, see [8]).

▶ **Theorem 11.** *There is a pushdown automaton with two states, such that there is no guarded sequential specification with the same process.*

Proof. Consider the example pushdown automaton in Figure 1. Suppose there is a finite guarded sequential specification with the same process, depicted by the representative in Figure 2. Without loss of generality we can assume that this specification is in Greibach Normal Form (see [8]). As a consequence, each state of the process graph generated by the automaton corresponds to a sequence of identifiers of the specification (as defined earlier). Take k a natural number that is larger than the number of process identifiers of the specification, take $i \leq k$ and consider the state $(\uparrow, 1^i)$ reached after executing i a-steps. From this state, consider any sequence of steps $\stackrel{w}{\longrightarrow\!\!\!\!\!\rightarrow}$ where $a \notin w$. Thus, w contains at most one c and at most i b's.

In the process graph generated by the recursive specification, this same sequence of steps $\stackrel{w}{\longrightarrow\!\!\!\!\!\rightarrow}$ is possible from the sequence of identifiers α_i bisimilar to state $(\uparrow, 1^i)$. Let X_i be the first element of α_i. From X_i, we can also execute at most one c-step and i b-steps, without executing an a-step.

Since k is larger than the number of process identifiers of the specification, there must be a repetition in the identifiers X_i $(i \leq k)$. Thus, there are numbers n, m, $n < m \leq k$, with $X_n = X_m$. The process identifier X_m can execute at most one c and $n < m$ b's without executing an a. But $\alpha_m \stackrel{b^m}{\longrightarrow\!\!\!\!\!\rightarrow}$, so the additional b-steps must come from the second and following identifiers of the sequence. As the second identifier is reached by just executing b's, this is a state reached by just executing a's and b's, so it must allow an initial c-step. Now we can consider $\alpha_m \stackrel{cb^m}{\longrightarrow\!\!\!\!\!\rightarrow}$. This sequence of steps must also reach the second identifier, but then, a second c can be executed, which is a contradiction.

Thus, our assumption was wrong, and the theorem is proved. ◀

We see that the contradiction is reached, because when we reach the second identifier in the sequence α_i, we do not know whether we are in a state relating to the initial state or the final state of the pushdown automaton. Going from the first identifier to the second identifier by means of the sequencing operator, no extra information can be passed along. We will describe a mechanism that allows the passing of extra information along the sequencing operator.

Theorem 11 holds for the sequencing operator we introduced, but it holds in the same way for the sequential composition operator of [6]. No intermediate acceptance is involved in the proof.

In the other direction, we can find a pushdown automaton with the same process as a given guarded sequential specification. The proof we give is more complicated than the classical proof, where it is only needed to find a pushdown automaton with the same language. There, we can handle all acceptance in a single state, which is only entered when the stack memory is empty. In bisimulation semantics, it is not true that every pushdown automaton is equivalent to one with such a single accepting state. We carefully need to consider every instance of intermediate acceptance. Consider the sequencing $(a.\mathbf{1} + \mathbf{1})\,;b.\mathbf{1}$. The first term in this sequencing shows intermediate acceptance, the second does not. As $(a.\mathbf{1} + \mathbf{1})\,;b.\mathbf{1} \leftrightarrows a.\mathbf{1}\,;b.\mathbf{1}$, the intermediate acceptance in the first term is redundant, and can be removed. We have to restrict the notion of Greibach normal form, in order to remove all redundant intermediate acceptance.

▶ **Definition 12.** *A guarded sequential specification is in Acceptance Irredundant Greibach normal form, AIGNF, if it is in Greibach normal form, and moreover, every state of the resulting process graph is given by a sequence of identifiers of the specification of the form $\alpha\beta$ $(\alpha, \beta \in \mathcal{P}^*)$, where all identifiers in α do not have intermediate acceptance, and all identifiers in β do have intermediate acceptance (or equal $\mathbf{1}$). The sequence α or β may be empty.*

It is proven in [8] that every guarded sequential specification can be transformed to one in Acceptance Irredundant Greibach normal form, so that all redundant intermediate acceptance is removed.

▶ **Theorem 13.** *For every guarded sequential specification there is a pushdown automaton with a bisimilar process graph, with at most two states.*

Proof. Let Δ be a guarded sequential specification over $\mathcal{P}$. Without loss of generality, we can assume Δ is in Acceptance Irredundant Greibach Normal Form [8]. Every state of the specification is given by a sequence of identifiers that is acceptance irredundant. The corresponding pushdown automaton has two states $\{n, t\}$. The initial state is n iff the initial identifier $S \not\downarrow$ and t iff the initial identifier $S \downarrow$ (as defined by the operational semantics), and the final state is t.

- For each summand $a.\alpha$ of an identifier X with $X \downarrow$ and the first identifier of α an identifier with $\downarrow$, add a step $t \xrightarrow{a[X/\alpha]} t$. Moreover, in case X is initial, a step $t \xrightarrow{a[\epsilon/\alpha]} t$;
- For each summand $a.\alpha$ of an identifier X with $X \downarrow$ and the first identifier of α an identifier with $\not\downarrow$, add a step $t \xrightarrow{a[X/\alpha]} n$. Moreover, in case X is initial, a step $t \xrightarrow{a[\epsilon/\alpha]} n$;
- For each summand $a.\alpha$ of an identifier X with $X \not\downarrow$ and the first identifier of α an identifier with $\downarrow$, add a step $n \xrightarrow{a[X/\alpha]} t$. Moreover, in case X is initial, a step $n \xrightarrow{a[\epsilon/\alpha]} t$;
- For each summand $a.\alpha$ of an identifier X with $X \not\downarrow$ and the first identifier of α an identifier with $\downarrow$, add a step $n \xrightarrow{a[X/\alpha]} n$. Moreover, in case X is initial, a step $n \xrightarrow{a[\epsilon/\alpha]} n$.

Now it is not difficult to check that the process of this pushdown automaton is the same as the process of the given guarded sequential specification. ◀

In the case of the sequential composition operator of [6], Theorem 13 also holds, the proof is simpler because of the more straightforward handling of intermediate acceptance, but it is not as simple as in the classical case.

6 Signals and conditions

In order to obtain the missing correspondence, we need a mechanism to pass state information along a sequencing operator. This mechanism is provided by propositional signals together with a conditional statement as given in [1], see also [6].

▶ **Definition 14.** *First of all, we add the data domain $\mathcal{B}$ of the* Booleans*, with two constants true and false, operators $\neg$ for negation and $\vee, \wedge$ for or, and. We use a set $P_1, ..., P_n$ as propositional variables. In this data type, we can make propositional formulas.*

Next, we can define a conditional statement or *guarded command*. Given a propositional formula ϕ, we write $\phi :\to x$, with the intuitive meaning '*if ϕ then x*'. In order to give an operational semantics, it is important to note that it is needed to know the values of the propositional variables in order to decide on possible transitions. Moreover, values of propositional variables can change during the execution of a process expression. Thus, we need a *valuation* that in each state of a process graph assigns *true* or *false* to each propositional variable. Upon executing an action a in a state with valuation v, a state with a possibly different valuation v' results. The resulting valuation v' is called the *effect* of the execution of a in a state with valuation v.

We present operational rules for guarded command in Fig. 5; it presupposes a function *effect* that associates with every action a and every valuation v its effect. We define when a term in a certain valuation can take a step or be in a final state.

$$\frac{}{\langle \mathbf{1}, v\rangle \downarrow} \qquad \frac{v' = \mathit{effect}(a, v)}{\langle a.x, v\rangle \xrightarrow{a} \langle x, v'\rangle}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle}{\langle x + y, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad \langle y + x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow}{\langle x + y, v\rangle \downarrow \quad \langle y + x, v\rangle \downarrow}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad v(\phi) = \mathit{true}}{\langle \phi :\to x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad v(\phi) = \mathit{true}}{\langle \phi :\to x, v\rangle \downarrow}$$

$$\frac{\langle x, v\rangle \downarrow \quad \langle y, v\rangle \downarrow}{\langle x \,;\, y, v\rangle \downarrow} \qquad \frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle}{\langle x \,;\, y, v\rangle \xrightarrow{a} \langle x' \,;\, y, v'\rangle}$$

$$\frac{\langle x, v\rangle \downarrow \quad \langle x, v\rangle \not\rightarrow \quad \langle y, v\rangle \xrightarrow{a} \langle y', v'\rangle}{\langle x \,;\, y, v\rangle \xrightarrow{a} \langle y', v'\rangle}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad X \stackrel{\text{def}}{=} x}{\langle X, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad X \stackrel{\text{def}}{=} x}{\langle X, v\rangle \downarrow}$$

Figure 5 Operational rules for guarded command ($a \in \mathcal{A} \cup \{\tau\}$).

On the basis of these rules, we can define a notion of bisimulation. We use *stateless* bisimulation, which means that two process graphs are bisimilar iff there is a bisimulation relation that relates two process expressions iff they are related under every possible valuation. The stateless bisimulation also allows non-determinism, as the effect of the execution of an action will allow every possible resulting sequel. See the example further on, after we also introduce the root signal operator. Notice that $\langle \mathit{false} :\to \mathbf{1}, v\rangle$ is not final for *any* valuation v.

Next, we introduce an operator that allows the observation of aspects of the current state of a process graph. The central assumption is that the visible part of the state of a process graph is a proposition, an expression over the booleans. We introduce the *root-signal emission operator* $\curvearrowright$. A term $\phi \curvearrowright x$ represents the process x that shows the signal ϕ in its initial state. In order to define this operator by operational rules, we need to define an additional predicate on terms, namely *consistency*. $\mathit{Cons}(\langle x, v\rangle)$ will not hold when the valuation of the root signal of x is false. A step $\xrightarrow{a}$ can only be between consistent states, and a state can only be final when it is consistent. Thus, the term $a.(\mathit{false} \curvearrowright x)$ can under no valuation execute action a.

The operational rules are defined in Fig. 6. First, we define the consistency predicate. Next, we find that the second, third, fourth and eighth rule of Table 5 require an extra condition. The other rules of Table 5 can remain unchanged. Finally, we give the operational rules of the root signal emission operator. To emphasise again the difference between guarded commands and root signal emission, term $\mathit{false} :\to (\mathbf{1} + a.\mathbf{1})$ is consistent and has, under any valuation, the same process graph as $\mathbf{0}$, whereas $\mathit{false} \curvearrowright (\mathbf{1} + a.\mathbf{1})$ is inconsistent, this state cannot be reached.

We again have a stateless bisimulation, where two terms are related iff any valuation that makes the root signal of one term *true* also makes the root signal of the other term *true* and for each such valuation, the process graphs of the terms are bisimilar.

The information given by the truth of the signals allows to determine the truth of some of the guarded commands. In this way, we can give a semantics for terms and specifications as regular process graphs, leaving out the valuations.

$$\frac{}{Cons(\langle \mathbf{0}, v\rangle)} \qquad \frac{}{Cons(\langle \mathbf{1}, v\rangle)} \qquad \frac{}{Cons(\langle a.x, v\rangle)}$$

$$\frac{Cons(\langle x, v\rangle)\ Cons(\langle y, v\rangle)}{Cons(\langle x + y, v\rangle)} \qquad \frac{Cons(\langle x, v\rangle)}{Cons(\langle \phi :\to x, v\rangle)} \qquad \frac{Cons(\langle x, v\rangle) \quad v(\phi) = true}{Cons(\langle \phi \mathbin{|\!>} x, v\rangle)}$$

$$\frac{Cons(\langle x, v\rangle) \quad \langle x, v\rangle \not\downarrow}{Cons(\langle x\,;y, v\rangle)} \qquad \frac{\langle x, v\rangle \downarrow \quad Cons(\langle y, v\rangle)}{Cons(\langle x\,;y, v\rangle)}$$

$$\frac{Cons(\langle x, v\rangle) \quad X \stackrel{\text{def}}{=} x}{Cons(\langle X, v\rangle)} \qquad \frac{Cons(\langle x, v'\rangle) \quad v' = effect(a, v)}{\langle a.x, v\rangle \xrightarrow{a} \langle x, v'\rangle}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad Cons(\langle y, v\rangle)}{\langle x + y, v\rangle \xrightarrow{a} \langle x', v'\rangle \ \ \langle y + x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad Cons(\langle y, v\rangle)}{\langle x + y, v\rangle \downarrow \ \ \langle y + x, v\rangle \downarrow}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad Cons(\langle x'\,;y, v'\rangle)}{\langle x\,;y, v\rangle \xrightarrow{a} \langle x'\,;y, v'\rangle}$$

$$\frac{\langle x, v\rangle \xrightarrow{a} \langle x', v'\rangle \quad v(\phi) = true}{\langle \phi \mathbin{|\!>} x, v\rangle \xrightarrow{a} \langle x', v'\rangle} \qquad \frac{\langle x, v\rangle \downarrow \quad v(\phi) = true}{\langle \phi \mathbin{|\!>} x, v\rangle \downarrow}$$

Figure 6 Operational rules for root-signal emission ($a \in \mathcal{A} \cup \{\tau\}$).

▶ **Definition 15.** *Let t, s be sequential terms with signals and conditions, let $a \in \mathcal{A} \cup \{\tau\}$ and suppose the root signal of t is not* false.

- $t \xrightarrow{a} s$ *iff for all valuations v such that $Cons(\langle t, v\rangle)$ we have $\langle t, v\rangle \xrightarrow{a} \langle s, effect(a, v)\rangle$,*
- $t \downarrow$ *iff for all valuations v such that $Cons(\langle t, v\rangle)$ we have $\langle t, v\rangle \downarrow$.*

The above definition makes all undetermined guarded commands *false*, as is illustrated in the following example.

▶ **Example 16.** Let P be a proposition variable, let a and b be actions without noticeable effect (i.e. $effect(a, v) = effect(b, v) = v$ for all valuations v), and let $s = P :\to a.\mathbf{1}$ and $t = P :\to b.\mathbf{1}$. Note that we have $Cons(\langle s, v\rangle)$ for all valuations v (since s does not emit any signal), but $\langle s, v\rangle \xrightarrow{a} \langle \mathbf{1}, v'\rangle$ only if $v(P) = true$. Hence, s does not have any outgoing transitions according to Definition 15. By the same reasoning, also t does not have any outgoing transitions. Therefore, s and t are bisimilar with respect to the transition relation induced on them by Definition 15.

When two terms are stateless bisimilar, then they are also bisimilar with respect to the transition relation induced on them by the Definition 15. The converse, however, does not hold: although the terms s and t in Example 16 are bisimilar, they are not stateless bisimilar.

To illustrate the interplay of root signal emission and guarded command, and to show how nondeterminism can be dealt with, we give the following example.

▶ **Example 17.** A coin toss can be described by the following term:

$$T \stackrel{\text{def}}{=} toss.(heads \mathbin{|\!>} \mathbf{1}) + toss.(tails \mathbin{|\!>} \mathbf{1}),$$

where $effect(toss, v) = v'$ is such that $v'(heads) = v'(tails) = true$ (for all v). Now, consider the expression

$$S \stackrel{\text{def}}{=} T\,;(heads :\to hurray.\mathbf{1} + tails :\to S).$$

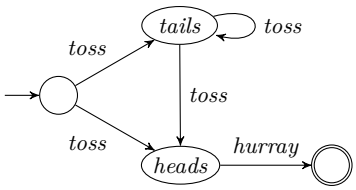


Figure 7 The process graph associated with the specification in Example 17.

Then S represents the process of tossing a coin until heads comes up, and its process graph is shown in Figure 7, as we shall now explain. Let

$$Heads = (heads \curvearrowright \mathbf{1})\,;(heads :\to hurray.\mathbf{1} + tails :\to S)$$

and let

$$Tails = (tails \curvearrowright \mathbf{1})\,;(heads :\to hurray.\mathbf{1} + tails :\to S)\ .$$

To see that $S \xrightarrow{toss} Tails$ and $S \xrightarrow{toss} Heads$, note that $\langle S, v\rangle \xrightarrow{toss} \langle Heads, effect(toss, v)\rangle$ and $\langle S, v\rangle \xrightarrow{toss} \langle Heads, effect(toss, v)\rangle$ for every valuation v.

To see that $Tails \xrightarrow{toss} Tails$ and $Tails \xrightarrow{toss} Heads$, first observe that $Cons(\langle Tails, v\rangle)$ if, and only if, $v(tails) = true$, and then note that for all such valuations v we, indeed, have $\langle Tails, v\rangle \xrightarrow{toss} \langle Tails, effect(toss, v)\rangle$ and $\langle Tails, v\rangle \xrightarrow{toss} \langle Heads, effect(toss, v)\rangle$.

It is instructive to see why we do *not* have that $Tails \xrightarrow{hurray} \mathbf{1}$. This is because if v is a valuation that satisfies $v(tails) = true$ and $v(heads) = false$, then we also have $Cons(\langle Tails, v\rangle)$, whereas $\langle Tails, v\rangle \overset{hurray}{\nrightarrow} \langle \mathbf{1}, effect(hurray, v)\rangle$.

Finally, to see that $Heads \xrightarrow{hurray} \mathbf{1}$, first observe that $Cons(\langle heads, v\rangle)$ if, and only if, $v(heads) = true$, and then note that, indeed, $\langle Heads, v\rangle \xrightarrow{hurray} \langle \mathbf{1}, effect(hurray, v)\rangle$ for all such valuations v.

7 The full correspondence

We prove that signals and conditions make it possible to find a guarded sequential specification with the same process as a given pushdown automaton.

▶ **Theorem 18.** *For every pushdown automaton there is a guarded sequential recursive specification with signals and conditions with the same process.*

Proof. Let $M = (\mathcal{S}, \mathcal{A}, \mathcal{D}, \to, \uparrow, \downarrow)$ be a pushdown automaton. We can assume M only has push and pop transitions. For every state $s \in \mathcal{S}$ we have a propositional variable $state(s)$. The *effect* function for every action invariantly results in a valuation that assigns *true* to every propositional variable $state(s)$, to make sure that the execution of an action from a consistent state always results in a consistent state. Note that it is the interplay between the emitted root signal and the guards that ensures appropriate continuations (cf. also Example 17). We proceed to define the recursive specification with initial identifier X and additional identifiers $\{X_d \mid d \in \mathcal{D}\}$.

- If M does not contain any transition of the form $\uparrow \xrightarrow{a[\epsilon/d]} s$, and the initial state is final, we can take $X \overset{\text{def}}{=} \mathbf{1}$ as the specification, and we do not need the additional identifiers.

- If M does not contain any transition of the form $\uparrow \xrightarrow{a[\epsilon/d]} s$, and the initial state is not final, we can take $X \stackrel{\text{def}}{=} \mathbf{0}$ as the specification, and we do not need the additional identifiers.
- Otherwise, there is some transition $\uparrow \xrightarrow{a[\epsilon/d]} s$ in M. For each such transition, add a summand

$$a.(state(s) \barwedge X_d \,;\, (state(\uparrow) :\rightarrow X + \neg state(\uparrow) :\rightarrow \mathbf{1}))$$

to the equation of X. The effect v of a in any valuation satisfies $v(state(s)) = true$. Besides these summands, add a summand $\mathbf{1}$ iff the initial state of M is final. Notice that no restriction on the initial valuation is necessary.
- Next, the equation for the added identifier X_d has a summand $state(s) :\rightarrow a.(state(t) \barwedge \mathbf{1})$ for each transition $s \xrightarrow{a[d/\epsilon]} t$, for every $s, t \in \mathcal{S}$. The effect v of a in any valuation satisfies $v(state(t)) = true$.
  In addition, the equation for the added identifier X_d has a summand

$$state(s) :\rightarrow a.((state(t) \barwedge X_e \,;\, X_d)$$

for each transition $s \xrightarrow{a[d/ed]} t$, for every $s, t \in \mathcal{S}$. The effect v of a in any valuation satisfies $v(state(t)) = true$.
Finally, the equation for the added identifier X_d has summands $state(s) :\rightarrow \mathbf{1}$ (whenever $s \in \downarrow$) or $state(s) :\rightarrow \mathbf{0}$ (otherwise). ◀

▶ **Example 19.** For the pushdown automaton in Fig. 1, we find the following guarded recursive specification:

$$S = a.(state\uparrow \barwedge A \,;(state\uparrow :\rightarrow S + state\downarrow :\rightarrow \mathbf{1})) + c.(state\downarrow \barwedge \mathbf{1})$$

$$A = state\downarrow :\rightarrow b.(state\downarrow \barwedge \mathbf{1}) +$$

$$+ state\uparrow :\rightarrow (a.(state\uparrow \barwedge A; A) + b.(state\uparrow \barwedge \mathbf{1}) + c.(state\downarrow \barwedge A)).$$

Finally, we are interested in the question whether the mechanism of signals and conditions is not *too* powerful: can we find a pushdown automaton with the same process for any guarded sequential specification with signals and conditions? We will show the answer is positive. This means we recover the perfect analogue of the classical theorem: the set of processes given by pushdown automata coincides with the set of processes given by guarded sequential specifications with signals and conditions.

In order to prove this final theorem, we need another refinement of Greibach normal forms, now in the presence of signals and conditions. We start out from a result from [6]. Actually, we modify this result, using the sequencing ; instead of the sequential composition $\cdot$. The proof goes the same way. We use identities $\chi \barwedge x \leftrightarrows \chi \barwedge \mathbf{0} + x$ and $\chi \barwedge \mathbf{0} + \psi :\rightarrow \mathbf{1} \leftrightarrows \chi \barwedge \mathbf{0} + (\chi \wedge \psi) :\rightarrow \mathbf{1}$.

▶ **Theorem 20.** *Every sequential term with signals and conditions is bisimilar to one in* head normal form*, i.e. of the form*

$$\chi \barwedge (\psi :\rightarrow \mathbf{1}) + \sum_{i=1}^{n} \phi_i :\rightarrow a_i.p_i \qquad a_i \in \mathcal{A} \cup \{\tau\}$$

for terms p_i, and is consistent in any state with a valuation v satisfying $v(\chi) = true$. Here, χ is the root signal *of the term, and $\chi \wedge \psi$ is the* acceptance condition *of the term. If the acceptance condition $\chi \wedge \psi = false$, we write this as*

$$\chi \mathbin{\triangleright} \mathbf{0} + \sum_{i=1}^{n} \phi_i :\to a_i.p_i \qquad a_i \in \mathcal{A} \cup \{\tau\}.$$

In the summation, we only write summands satisfying $\chi \wedge \phi_i \neq false$.

We see a term in head normal form is accepting in any state with a valuation v that makes the acceptance condition *true*, and it shows intermediate acceptance whenever the sum is nonempty and there is i with $v(\chi \wedge \psi \wedge \phi_i) = true$.

Based on this, we can write each equation in a guarded sequential specification with signals and conditions in a Greibach normal form, as follows:

$$X = \chi \mathbin{\triangleright} (\psi :\to \mathbf{1}) + \sum_{i=1}^{n} \phi_i :\to a_i.\alpha_i \qquad a_i \in \mathcal{A} \cup \{\tau\}, X \in \mathcal{P}, \alpha_i \in \mathcal{P}^*.$$

Here, χ is the root signal of X, and ψ the acceptance condition of X, writing again

$$X = \chi \mathbin{\triangleright} \mathbf{0} + \sum_{i=1}^{n} \phi_i :\to a_i.\alpha_i \qquad a_i \in \mathcal{A} \cup \{\tau\}, X \in \mathcal{P}, \alpha_i \in \mathcal{P}^*$$

if the acceptance condition is *false*. We define the Acceptance Irredundant Greibach normal form as before, disregarding the remaining signals and conditions.

▶ **Theorem 21.** *For every guarded sequential recursive specification with signals and conditions there is a pushdown automaton with the same process.*

Proof. Suppose a finite guarded sequential specification with signals and conditions is given. Without loss of generality we can assume this specification is in Acceptance Irredundant Greibach normal form, so every state of the specification is given by a sequence of identifiers that is acceptance irredundant. Consider the set of propositional variables $P_1, ..., P_n$ occurring in this specification. For each possible valuation $v : \{P_1, ..., P_n\} \to \{true, false\}$, we create two states in the pushdown automaton to be constructed, the state $\langle v, t\rangle$ is final and the state $\langle v, n\rangle$ is not. Then, we go along the lines of Theorem 13.

Take a valuation v such that the root signal of the initial identifier is *true*. If $v(\psi) = true$, where ψ is the acceptance condition of the initial identifier, then the initial state is $\langle v, t\rangle$. Otherwise, the initial state is $\langle v, n\rangle$. Next, for any identifier X of the specification with root signal χ and acceptance condition ψ, and any valuation v satisfying $v(\chi) = true$, we consider two cases.

- *Case 1.* Suppose $v(\psi) = true$. Look at a summand $\phi :\to a.\alpha$ of X. If $v(\phi) = false$ or the effect of executing a from v, the valuation v', makes the root signal of the first identifier of α *false*, we add no steps; otherwise, we consider two subcases.
  - *Subcase 1.a.* Suppose v' makes all root signals of the identifiers α and all acceptance conditions of the identifiers α *true*. Add a step $\langle v, t\rangle \xrightarrow{a[X/\alpha]} \langle v', t\rangle$. Moreover, in case X is initial, a step $\langle v, t\rangle \xrightarrow{a[\epsilon/\alpha]} \langle v', t\rangle$;
  - *Subcase 1.b.* Suppose v' makes some root signal of α or some acceptance condition of α *false*. Add a step $\langle v, t\rangle \xrightarrow{a[X/\alpha]} \langle v', n\rangle$. Moreover, in case X is initial, a step $\langle v, t\rangle \xrightarrow{a[\epsilon/\alpha]} \langle v', n\rangle$;

Next, repeat this procedure for the remaining summands of X.

- *Case 2.* Suppose $v(\psi) = \mathit{false}$. Look at a summand $\phi :\to a.\alpha$ of X. If $v(\phi) = \mathit{false}$ or the effect of executing a from v, the valuation v' makes the root signal of the first identifier of α *false*, we add no steps; otherwise, we consider two subcases.
  - *Subcase 2.a.* Suppose v' makes all root signals of α and all acceptance conditions of α *true*. Add a step $\langle v, n\rangle \xrightarrow{a[X/\alpha]} \langle v', t\rangle$. Moreover, in case X is initial, a step $\langle v, n\rangle \xrightarrow{a[\epsilon/\alpha]} \langle v', t\rangle$;
  - *Subcase 2.b.* Suppose v' makes some root signal of α or some acceptance condition of α *false*. Add a step $\langle v, n\rangle \xrightarrow{a[X/\alpha]} \langle v', n\rangle$. Moreover, in case X is initial, a step $\langle v, n\rangle \xrightarrow{a[\epsilon/\alpha]} \langle v', n\rangle$;

  Next, repeat this procedure for the remaining summands of X.

Now it is not difficult to check that the process of this pushdown automaton coincides with the process of the given guarded sequential specification. ◀

8 Conclusion

We looked at the classical theorem, that the set of languages given by a pushdown automaton coincides with the set of languages given by a context-free grammar. A language is an equivalence class of process graphs modulo language equivalence. A process is an equivalence class of process graphs modulo bisimulation. The set of processes given by a pushdown automaton coincides with the set of processes given by a finite guarded sequential recursive specification, if and only if we add a notion of state awareness, that allows to pass on some information during sequencing.

We see that signals and conditions add expressive power to TSP, since a signal can be passed along the sequencing operator. If we go to the theory BCP, so without sequencing but with parallel composition, then we know from [1] that value passing can be replaced by signal observation. We leave it as an open problem, whether or not signals and conditions add to the expressive power of BCP.

This paper contributes to our ongoing project to integrate automata theory and process theory. As a result, we can present the foundations of computer science using a computer model with interaction. Such a computer model relates more closely to the computers we see all around us.

References

1 J. C. M. Baeten and J. A. Bergstra. Process algebra with propositional signals. *Theor. Comput. Sci.*, 177(2):381–405, 1997. doi:10.1016/S0304-3975(96)00253-8.

2 J. C. M. Baeten, P. J. L. Cuijpers, B. Luttik, and P. J. A. van Tilburg. A process-theoretic look at automata. In F. Arbab and M. Sirjani, editors, *Fundamentals of Software Engineering, Third IPM International Conference, FSEN 2009, Kish Island, Iran, April 15-17, 2009, Revised Selected Papers*, volume 5961 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2009. doi:10.1007/978-3-642-11623-0_1.

3 J. C. M. Baeten, P. J. L. Cuijpers, and P. J. A. van Tilburg. A context-free process as a pushdown automaton. In F. van Breugel and M. Chechik, editors, *Proceedings CONCUR'08*, number 5201 in Lecture Notes in Computer Science, pages 98–113, 2008.

4 J. C. M. Baeten, B. Luttik, and P. J. A. van Tilburg. Reactive Turing machines. *Information and Computation*, 231:143–166, 2013. Fundamentals of Computation Theory. doi:10.1016/j.ic.2013.08.010.

5 J. C. M. Baeten, B. Luttik, and F. Yang. Sequential composition in the presence of intermediate termination (extended abstract). In K. Peters and S. Tini, editors, *Proceedings Combined 24th International Workshop on Expressiveness in Concurrency and 14th Workshop on Structural Operational Semantics, EXPRESS/SOS 2017, Berlin, Germany, 4th September 2017.*, volume 255 of *EPTCS*, pages 1–17, 2017. doi:10.4204/EPTCS.255.1.

6 J. C.M. Baeten, A. A. Basten, and M. A. Reniers. *Process algebra: equational theories of communicating processes*, volume 50. Cambridge university press, 2010.

7 A. Belder. Decidability of bisimilarity and axiomatisation for sequential processes in the presence of intermediate termination. Master's thesis, Eindhoven University of Technology, 2018. Available from https://research.tue.nl/en/studentTheses/decidability-of-bisimilarity-and-axiomatisation-for-sequential-pr.

8 A. Belder, B. Luttik, and J. C. M. Baeten. Sequencing and intermediate acceptance: axiomatisation and decidability of bisimilarity. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science, CALCO 2019*, Leibniz International Proceedings in Informatics, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CALCO.2019.11.

9 R. N. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *J. ACM*, 43(5):863–914, 1996. doi:10.1145/234752.234756.

10 D. Goldin and P. Wegner. The interactive nature of computing: Refuting the strong Church-Turing thesis. *Minds and Machines*, 18(1):17–38, 2008.

11 J. F. Groote. Transition system specifications with negative premises. *Theor. Comput. Sci.*, 118(2):263–299, 1993. doi:10.1016/0304-3975(93)90111-6.

12 B. Luttik. Divergence-preserving branching bisimilarity. In O. Dardha and J. Rot, editors, *Proceedings Combined 27th International Workshop on Expressiveness in Concurrency and 17th Workshop on Structural Operational Semantics, EXPRESS/SOS 2020, Online, 31 August 2020*, volume 322 of *EPTCS*, pages 3–11, 2020. doi:10.4204/EPTCS.322.2.

13 R. J. van Glabbeek. The meaning of negative premises in transition system specifications II. *J. Log. Algebr. Program.*, 60-61:229–258, 2004. doi:10.1016/j.jlap.2004.03.007.

14 R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996. doi:10.1145/233551.233556.

15 C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nord. J. Comput.*, 2(2):274–302, 1995.

16 P. Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):80–91, 1997.

From Farkas' Lemma to Linear Programming: an Exercise in Diagrammatic Algebra

Filippo Bonchi
University of Pisa, Italy

Alessandro Di Giorgio
University of Pisa, Italy

Fabio Zanasi
University College London, UK

Abstract

Farkas' lemma is a celebrated result on the solutions of systems of linear inequalities, which finds application pervasively in mathematics and computer science. In this work we show how to formulate and prove Farkas' lemma in diagrammatic polyhedral algebra, a sound and complete graphical calculus for polyhedra. Furthermore, we show how linear programs can be modeled within the calculus and how some famous duality results can be proved.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases String diagrams, Farkas Lemma, Duality, Linear Programming

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.9

Category (Co)algebraic pearls

Funding *Filippo Bonchi*: Supported by the Ministero dell'Università e della Ricerca of Italy under Grant No. 201784YSZ5, PRIN2017 – ASPRA (Analysis of Program Analyses).
Alessandro Di Giorgio: Supported by the Ministero dell'Università e della Ricerca of Italy under Grant No. 201784YSZ5, PRIN2017 – ASPRA (Analysis of Program Analyses).
Fabio Zanasi: Supported by EPSRC EP/V002376/1.

1 Introduction

Farkas' lemma is a classical result on the solutions of systems of linear inequalities, which appears ubiquitously across various fields of Mathematics and Computer Science; more than a century after its introduction in [16, 17], it continues to receive attention and generate new lines of research [3, 10, 15, 22, 30, 25, 31, 4, 24, 28, 1]. Throughout the decades, different proofs have been given, and many variations have been proposed. The most established formulation asserts that, given an $m \times n$ matrix A, a vector $b \in \mathbb{R}^m$ and their transposes A^T and b^T, exactly one of the following two propositions is true.

$$\text{(a) } \exists x \in \mathbb{R}^n \text{ s.t. } x \geq 0 \text{ and } Ax = b \qquad \text{(b) } \exists y \in \mathbb{R}^m \text{ s.t. } A^T y \geq 0 \text{ and } b^T y < 0$$

Farkas' lemma finds application in a number of different scenarios, ranging from non-linear optimisation [28, 4] to the algebraic semantics of non-deterministic and probabilistic systems [23]. Most computer scientists first meet Farkas' lemma when studying duality theory in linear programming. A gentle introduction to this theory is provided by the *farmer problem*.

A farmer grows wheat and barley on a land of size l, with a provision f of fertilizer and p of pesticide. To grow one unit of wheat the farmer needs one unit of land, f_1 units of fertilizer and p_1 units of pesticide. Analogously, one unit of barley requires one unit of land, f_2 of fertilizer and p_2 of pesticide. The sell prices for wheat and barley are, respectively, s_1 and

s_2. By fixing x_1 to be the units of wheat and x_2 those of barley to be produced, the farmer should solve the following linear program to maximize the profit out of the production.

$$max\{c \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mid x_1, x_2 \geq 0,\ A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq b\} \text{ where } c = \begin{pmatrix} s_1 & s_2 \end{pmatrix},\ A = \begin{pmatrix} 1 & 1 \\ f_1 & f_2 \\ p_1 & p_2 \end{pmatrix},\ b = \begin{pmatrix} l \\ f \\ p \end{pmatrix}$$

Now assume that a planning board needs to establish prices for land, fertilizer and pesticide. The board's job is to minimize the cost of production while assuring some profit to the farmer. To do so, it is sufficient to solve the following program where A, b and c are as above.

$$min\{b^T \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \mid y_1, y_2, y_3 \geq 0,\ A^T \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \leq c^T\}$$

The problem of the farmer and the one of the board are a typical example of a pair of dual problems. A result in duality theory (which makes the relevance of Farkas' lemma apparent) is that, if a problem has unbounded solution, then its dual has no solution. Most importantly, when a problem and its dual have finite solutions, then these solutions coincide. In the example above, the minimum cost of the production and the maximum profit of the farmer should then be equal.

In this paper we revisit Farkas' lemma and duality results in linear programming through the lens of *string diagrams*.

String diagrams are a graphical syntax for representing arrows of symmetric monoidal categories [33]. In recent years, increasingly they have been adopted as a formal language to study component-based systems across different fields of science [12, 2, 18, 19, 21, 29, 32] using the compositional methods that are typical of programming language semantics. One striking property of this approach is that, even though string diagrams have an appealing graphical representation, they are completely formal syntactic objects. Furthermore, they may receive semantics interpretation in some mathematical domain (such as functions, relations, matrices, subspaces, etc.) and many results have been provided on how equational theories of string diagrams are able to *axiomatise* semantic equality over these domains, see e.g. [6, 8, 36, 37, 2, 7]. Such a *complete* equational theory yields a powerful pictorial calculus to reason algebraically about system behaviour, for instance in concurrency [6, 11], control [9, 2] and quantum theory [13].

The core of the calculus that we exploit in this paper is the theory of Interacting Hopf Algebras [36, 8, 2], originally introduced to reason about the behaviour of signal flow graphs [34]. Such theory has been extended first in [7] to study non-passive electrical network and concurrent connectors [11], and then in [5], for studying continuous Petri nets [14]. The latter extension, called *diagrammatic polyhedral algebra*, provides a sound and complete calculus which is able to express exactly *polyhedra*. We claim this is the proper string diagrammatic setting to express Farkas' lemma and duality in linear programming.

In diagrammatic polyhedral algebra, recalled in Section 2, different entities of traditional algebra, like vectors, matrices and subsets $C \subseteq \mathbb{R}^n$ are all regarded as relations amongst vectors spaces. Starting from few primitive relations (depicted as wires and gates of circuits), one can syntactically construct all polyhedra by means of relational composition and cartesian product (graphically rendered as horizontal and vertical juxtaposition). It is exactly this linguistic aspect the main novelty of our proof of Farkas' lemma: statements about existence of solutions, like (a) and (b) above, translate into equations amongst terms of the string diagrammatic syntax; proofs are symbolic manipulation of diagrams, whose soundness is

guaranteed by the axiomatisation. Moreover, compositionality allows to break complex notions into simple inductive definitions on the sets of primitive relations. For instance, the *polar operator* which is given inductively in Section 3, captures the notions of polar and dual cone that are defined in the traditional language by mean of universal quantifications.

In the context of diagrammatic polyhedral algebra, the proof of Farkas' lemma becomes straightforward: using a basic observation, named the *lemma of alternatives* in Section 4, the proof –in Section 5– reduces to compute the polar operator over a certain string diagram.

The final part of our work (Section 6) is dedicated to duality in linear programming. Interestingly, diagrammatic polyhedral algebra allows to prove various duality theorems in a rather different way than those found in traditional textbooks (see e.g. [35]). In the classical approach, one first needs to massage the dual problems to bring them into an appropriate shape, and then prove, in sequence, a weak and a strong duality theorems. Our proof method instead is based on a general principle (Theorem 23) that, independently from the shape of the problem at hand, allows to prove all the results at once. Curiously, our proof does not rely on Farkas' lemma: rather both the duality theorems and Farkas' lemma stem from general results encoded in the axiomatisation of diagrammatic polyhedral algebra.

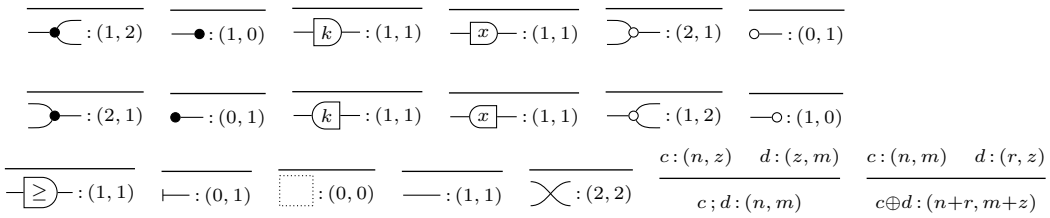
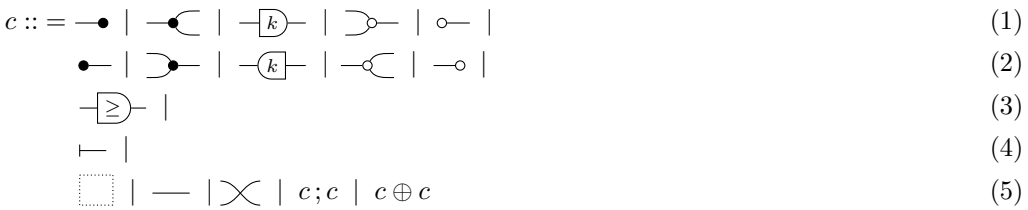


Figure 1 Sort inference rules.

2 Diagrammatic Polyhedral Algebra

This section presents a calculus of string diagrams for reasoning about polyhedra, which we will later use to prove Farkas' Lemma and the duality theorems for linear programming. The calculus was first introduced in [5], to which we refer for a more detailed exposition.

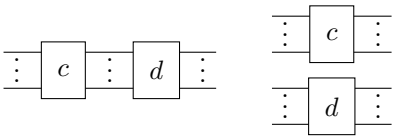
We fix an *ordered field* k, i.e. a field equipped with a total order $\geq$ such that for all $i, j, k \in \mathsf{k}$: (a) if $i \geq j$, then $i + k \geq j + k$; (b) if $i \geq 0$ and $j \geq 0$, then $i \cdot j \geq 0$. The syntax of the calculus is given by the following context free grammar, where k ranges over k.



We shall consider only terms that are *sortable*, i.e. that one may associate with a pair (n, m) of natural numbers $n, m \in \mathbb{N}$ using the rules in Figure 1.

The above syntax specification purposefully uses a graphical rendering of the components. As customary for string diagrams, we will render composition via ; and $\oplus$ graphically by

horizontal and vertical juxtaposition of boxes, respectively.

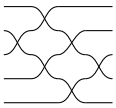


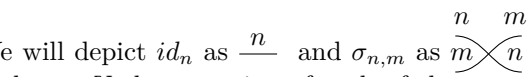
For an example, consider the diagram c in Example 4 below. This represents the term $(\multimap \oplus \multimap \oplus \bullet\multimap);(\multimap \oplus \boxed{k_2} \oplus \multimap \oplus \square);(\boxed{k_1} \oplus \multimap \oplus \supset\!\circ \oplus \bullet\multimap)$.

Note that one-dimensional syntax coincides with diagrammatic notation only modulo certain structural rules (e.g. associativity of composition), which amount to the equations of *symmetric monoidal categories* [33] (SMCs). It turns out that structurally equivalent terms have the same meaning in the semantic model we will consider below. Thus, henceforth we shall exclusively focus on string diagrams as our notation for syntax.

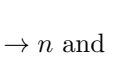
It is worth to also recall the categorical viewpoint on diagrammatic syntax. Equivalently to the presentation given above, one may formalise string diagrams as the morphisms of a *prop* (product and permutation category [27, 26]), i.e. a strict SMC with objects the natural numbers, where $\oplus$ on objects is by addition. We introduce the prop for our syntax below.

▶ **Definition 1.** *The prop freely generated by* (1), (2), (3) *and* (4) *is denoted by* PDiag. *In other words,* PDiag *is the prop where arrows $n \to m$ are terms of sort (n, m) quotiented by the axioms of symmetric monoidal categories. Composition* ; *and monoidal product* $\oplus$ *of diagrams are given by the syntax operations in* (5). *The identities are* $id_0 :=$ *and* $id_{n+1} := id_n \oplus \multimap$. *The symmetries* $\sigma_{n,m} \colon n + m \to m + n$ *are defined in the obvious way starting from* $\sigma_{1,1} :=$  . *For instance,* $\sigma_{2,3}$ *is the diagram below.*

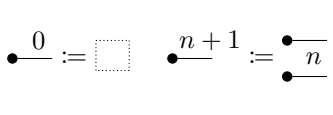


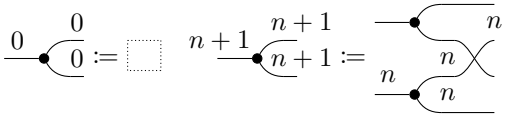
We will depict id_n as  and $\sigma_{n,m}$ as 

. Using these diagrams one can define for each $n \in \mathbb{N}$ the n-version of each of the generator in (1), (2), (3) and (4). For instance,

$\colon 0 \to n$ and 

$\colon n \to n + n$ are inductively defined as





When clear from the context, we will omit the n. A semantic interpretation for string diagrams of PDiag will be provided by morphisms in another prop, which we present below.

▶ **Definition 2.** Rel_k *is the prop where arrows $n \to m$ are relations $R \subseteq \mathsf{k}^n \times \mathsf{k}^m$.*

- *Composition is relational: given $R \colon n \to m$ and $S \colon m \to o$,*

$$R \,;\, S = \{\, (u, v) \in \mathsf{k}^n \times \mathsf{k}^o \;\mid\; \exists w \in \mathsf{k}^m.\ (u, w) \in S \wedge (w, v) \in R \,\}$$

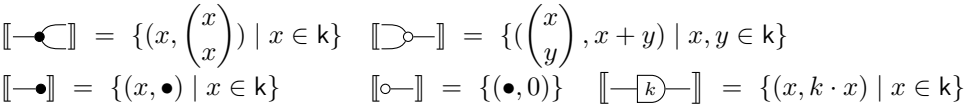
- *The monoidal product is cartesian product: given $R \colon n \to m$ and $S \colon o \to p$,*

$$R \oplus S = \{\, (\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}) \in \mathsf{k}^{n+o} \times \mathsf{k}^{m+p} \;\mid\; (u_1, v_1) \in R \wedge (v_1, v_2) \in S \,\}$$

- *The symmetries* $\sigma_{n,m} \colon n+m \to m+n$ *are the relations*

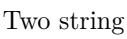
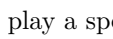
$$\{ (\begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} v \\ u \end{pmatrix}) \mid u \in \mathsf{k}^n, v \in \mathsf{k}^m \}$$

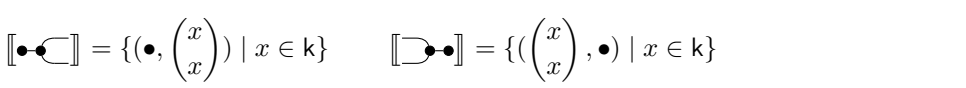
We can now formally define the semantic interpretation as a *prop morphism* (an identity-on-objects symmetric monoidal functor) $[\![\cdot]\!] : \mathsf{PDiag} \to \mathsf{Rel}_\mathsf{k}$. For the generators in (1), $[\![\cdot]\!]$ is



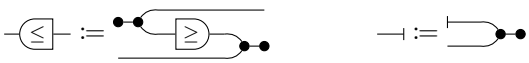
(6)

and, symmetrically, for the generators in (2). For instance, $[\![-(k)-]\!] = \{(k \cdot x, x) \mid x \in \mathsf{k}\}$. For the generators in (3) and (4), the semantics is defined, respectively, as $[\![-(\geq)-]\!] = \{ (x, y) \mid x, y \in \mathsf{k}, x \geq y \}$ and $[\![\vdash]\!] = \{(\bullet, 1)\}$. The semantics of the identities, symmetries and compositions – in (5)– is given by the *functoriality* of $[\![\cdot]\!]$, e.g., $[\![c \,;\, d]\!] = [\![c]\!] \,;\, [\![d]\!]$ and $[\![\square]\!] = [\![id_0]\!] = \{(\bullet, \bullet)\}$. Above we used $\bullet$ for the unique element of the vector space k^0.

▶ **Example 3.** Two string diagrams will play a special role in our exposition:  and  . By definition of $[\![\cdot]\!]$, note that their semantics forces the two ports on the right (resp. left) to carry the same value, thus acting as a left (right) feedback.



We can use these feedback diagrams to arbitrarily move wires from left to right. For instance



As expected, $[\![-(\leq)-]\!] = \{(y, x) \mid x, y \in \mathsf{k},\, x \geq y\}$ and $[\![\dashv]\!] = \{(1, \bullet)\}$.

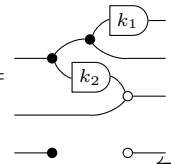
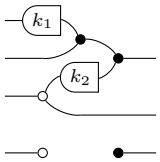
In [5] it is shown that diagrams of PDiag can express, amongst all the relations $R \subseteq \mathsf{k}^n \times \mathsf{k}^m$, exactly all those that are *polyhedra*, *cf.* Example 7 below. Moreover, it is worth recalling that fragments of PDiag also characterise well-known classes of relational objects, as indicated in the table below (see [36, 5] for an overview of these results).

prop	**syntax**	**semantics**
$\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$	(1), (5)	matrices
$\mathsf{MD}^{\leftarrow}_{\mathsf{iag}}$	(2), (5)	reversed matrices
LDiag	(1), (2), (5)	linear relations (sub-spaces)
PCDiag	(1), (2), (3), (5)	polyhedral cones
PDiag	(1), (2), (3), (4), (5)	polyhedra

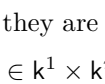
(7)

For instance, the arrows of PDiag, which are only built from the components in (1) and (5), form a sub-prop of PDiag, denoted by $\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$, and characterise k-matrices – in terms of the semantics functor $[\![\cdot]\!] : \mathsf{PDiag} \to \mathsf{Rel}_\mathsf{k}$, they denote precisely the relations of the form $\{(x, Ax) \mid x \in \mathsf{k}^p\}$ for some matrix A. Similarly $\mathsf{MD}^{\leftarrow}_{\mathsf{iag}}$, LDiag and PCDiag are the sub-props of PDiag of arrows built from the generators specified in (7). Hereafter we illustrate some examples of these fragments, and the corresponding semantic characterisation.

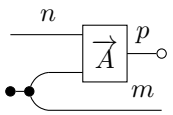
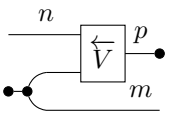
▶ **Example 4** ((Reversed) Matrices). As mentioned, diagrams $c\colon n \to m$ in $\mathsf{MD}\overrightarrow{\mathsf{iag}}$ denote precisely the $m \times n$ matrices (see [36] for all details). Consider for instance, the diagram $c\colon 3 \to 4$ and its representation as a 4×3 matrix. Note that $A_{ij} = k$ whenever k is the scalar encountered on the path from the ith port to the jth port. If there is no path, then $A_{ij} = 0$. It is easy to check that $[\![c]\!] = \{(x, y) \in \mathsf{k}^3 \times \mathsf{k}^4 \mid y = Ax\}$.

$c =$  $A = \begin{pmatrix} k_1 & 0 & 0 \\ 1 & 0 & 0 \\ k_2 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ $d =$ 

Dually, diagrams in $\mathsf{MD}\overleftarrow{\mathsf{iag}}$ are "reversed" matrices: inputs on the right and outputs on the left. For instance $d\colon 4 \to 3$ again encodes A, but its semantics is $[\![d]\!] = \{(y, x) \in \mathsf{k}^4 \times \mathsf{k}^3 \mid y = Ax\}$.

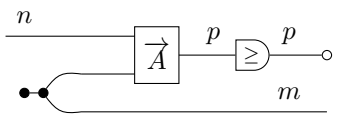
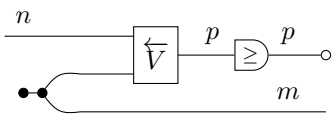
Hereafter we will use  and  for some diagrams of $\mathsf{MD}\overrightarrow{\mathsf{iag}}$ and, respectively $\mathsf{MD}\overleftarrow{\mathsf{iag}}$, corresponding to some $m \times n$ matrix A. For matrices of type $m \times 1$ and $1 \times n$ we will use lower case letters, usually b and c respectively. It is worth remarking that while $m \times 1$ matrices and vectors in k^m have the same representation in the traditional notation, in PDiag, they are presented as  and  . Indeed, the semantics of the former is $\{(k, bk) \in \mathsf{k}^1 \times \mathsf{k}^m \mid k \in \mathsf{k}\}$, while the semantics of the latter is $\{(\bullet, b) \in \mathsf{k}^0 \times \mathsf{k}^m\}$.

▶ **Example 5** ($\mathsf{Linear\ Relations}$). Consider the following diagrams in LDiag.

 (8)  (9)

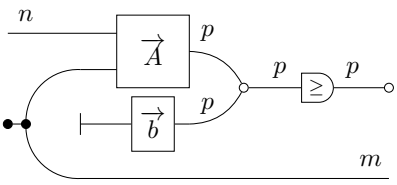
It easy to check that the semantics of (8) is the set $\{(x, y) \in \mathsf{k}^n \times \mathsf{k}^m \mid A \begin{pmatrix} x \\ y \end{pmatrix} = 0\}$, that is the set of solutions of some system of linear equations. Such system has p rows in $n + m$ variables: n variables stand on the left and m variables on the right. This means that $[\![(8)]\!]$ is a sub-vector space of $\mathsf{k}^n \times \mathsf{k}^m$, namely a *linear relation*. The semantics of (9) is $\{(x, y) \in \mathsf{k}^n \times \mathsf{k}^m \mid \exists z \in \mathsf{k}^p \text{ s.t. } \begin{pmatrix} x \\ y \end{pmatrix} = Vz\}$, that is the linear hull of the set of column vectors of the matrix V, or in other words the subspace generated by V. Recall that any subspace can be represented both in the form of a system of linear equations and in the form of a set of generating vectors. Indeed, diagrams (8) and (9) represents two normal forms for the diagrams in LDiag.

▶ **Example 6** ($\mathsf{Polyhedral\ cones}$). Consider the following diagrams in PCDiag

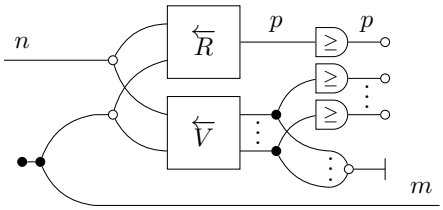
 (10)  (11)

with semantics $\{(x, y) \in \mathsf{k}^n \times \mathsf{k}^m \mid A \begin{pmatrix} x \\ y \end{pmatrix} \geq 0\}$ and $\{(x, y) \in \mathsf{k}^n \times \mathsf{k}^m \mid \exists z \in \mathsf{k}^p \text{ s.t. } \begin{pmatrix} x \\ y \end{pmatrix} = Vz, z \geq 0\}$, respectively. The semantics of (10) is thus the set of solutions of a systems of linear *in*equalities, namely a *polyhedral cone*, while the semantics of (11) is the conic hull of V (seen as a set of column vectors). Similarly to Example 5, diagrams in (10) and (11) can be regarded as two normal forms for diagrams in PCDiag.

▶ **Example 7** (Polyhedra). Consider the following diagrams in PDiag.



(12)



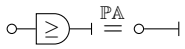
(13)

It is easy to check that the semantics of (12) is the relation $\{\,(x,y) \in \mathsf{k}^n \times \mathsf{k}^m \mid A\begin{pmatrix}x\\y\end{pmatrix} + b \geq 0\,\}$ and thus the representation of a *polyhedron* as the set of solutions of a system of affine inequalities. The semantics of (13) is the relation $\{(x,y) \in \mathsf{k}^n \times \mathsf{k}^m \mid \exists z \in \mathsf{k}^p, w \in \mathsf{k}^o \text{ s.t. } z \geq 0, w \geq 0, \sum w_i = 1, Rz + Vw = \begin{pmatrix}x\\y\end{pmatrix}\}$ and thus a *vertex* representation of a polyhedron. In other words, $[\![(13)]\!]$ is Minkowsky sum of the conic hull of R, $\{\begin{pmatrix}x\\y\end{pmatrix} \mid \exists z \in \mathsf{k}^p, \text{ s.t. } z \geq 0, Rz = \begin{pmatrix}x\\y\end{pmatrix}\}$, and of the *convex hull* of V, $\{\begin{pmatrix}x\\y\end{pmatrix} \mid \exists w \in \mathsf{k}^o \text{ s.t. } w \geq 0, \sum w_i = 1, Vw = \begin{pmatrix}x\\y\end{pmatrix}\}$.

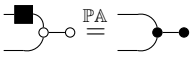
The functor $[\![\cdot]\!] : \mathsf{PDiag} \to \mathsf{Rel}_\mathsf{k}$ is not *faithful*: two different string diagrams may denote the same relation. However, PDiag can be equipped with a *sound and complete axiomatisation*, meaning an equational theory making two diagrams c and d equal precisely when $[\![c]\!] = [\![d]\!]$. Such axiomatisation, called *Polyhedral Algebra* ($\mathbb{PA}$) is illustrated in Figure 2, where we write $l = r$ for the two inequalities $l \sqsubseteq r$ and $r \sqsubseteq l$. In order to state the completeness theorem, we define $\overset{\mathbb{PA}}{\sqsubseteq}$ as the smallest precongruence containing all the pairs (c,d) such that $c \sqsubseteq d$ appears in the Figure 2. In other words, $\overset{\mathbb{PA}}{\sqsubseteq}$ is the smallest relation containing $\sqsubseteq$ which is closed by reflexivity, transitivity, composition ; and monoidal product $\otimes$. Finally, we write $c \overset{\mathbb{PA}}{=} d$ iff $c \overset{\mathbb{PA}}{\sqsubseteq} d$ and $d \overset{\mathbb{PA}}{\sqsubseteq} c$.

▶ **Theorem 8** (From [5]). *For all diagrams c, d in* PDiag*, $[\![c]\!] \subseteq [\![d]\!]$ if and only if $c \overset{\mathbb{PA}}{\sqsubseteq} d$.*

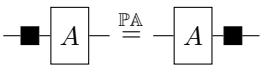
Here are some interesting consequences of the theory $\mathbb{PA}$, where we use —■— for —[−1]—.



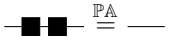
(14)



(15)



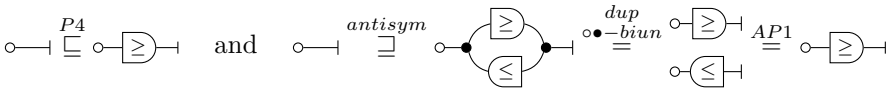
for any A in LDiag (16)



(17)

Theorem 8 implies that equivalences like (14), (15) and (17) may be also proved by purely graphical means, using derivations involving the axioms of $\mathbb{PA}$, without resorting to the semantic interpretation $[\![\cdot]\!]$. The proofs of more sophisticated statements, as (16), involve axioms in combination with other proof techniques, e.g., induction.

The following is an example of derivation proving (14).



(18)

Note that in (18) we used a version of axioms $P4$, *dup* and $AP1$ where diagrams are "rotated over the y axis". We formalise such a notion, in a way that justifies this use.

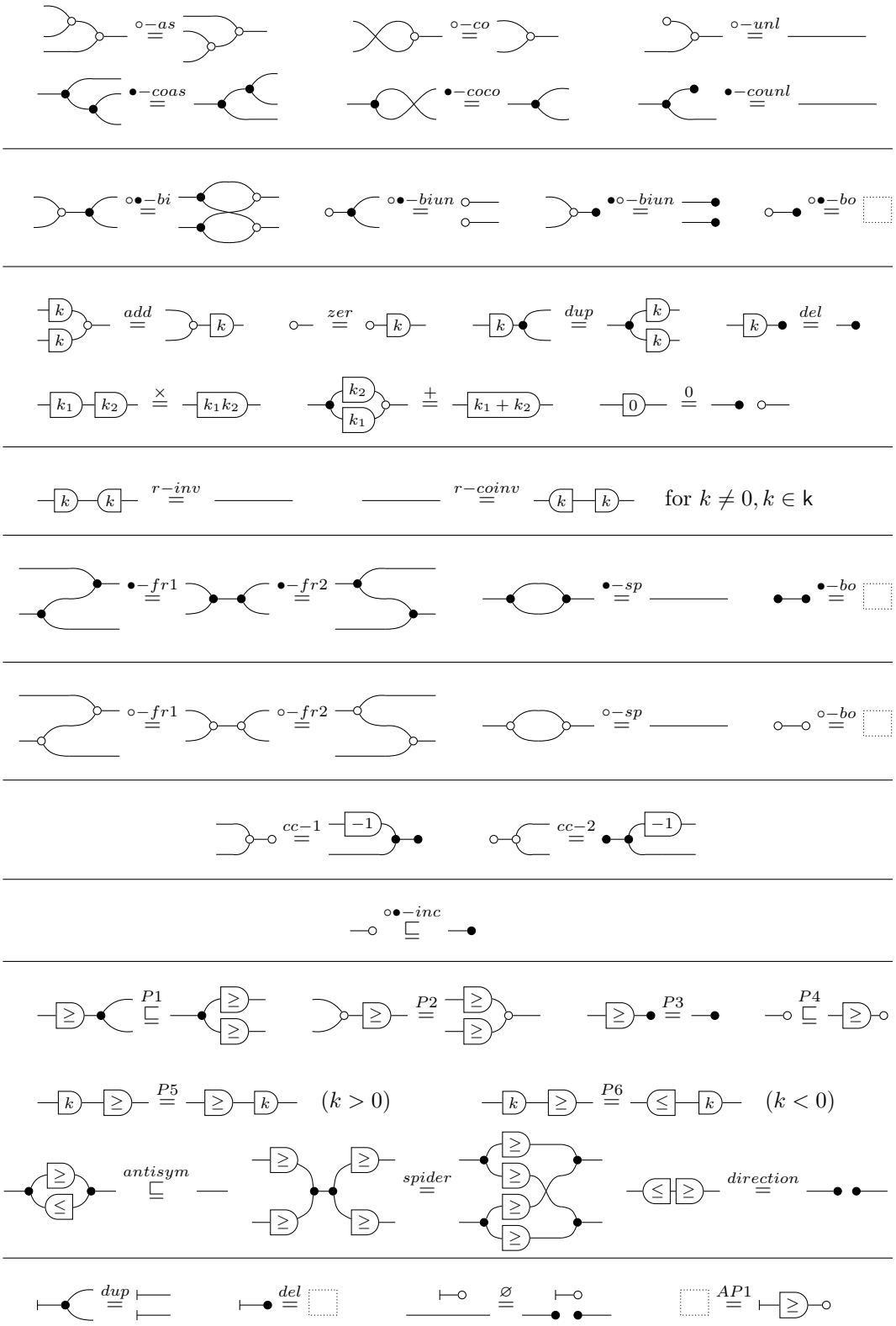
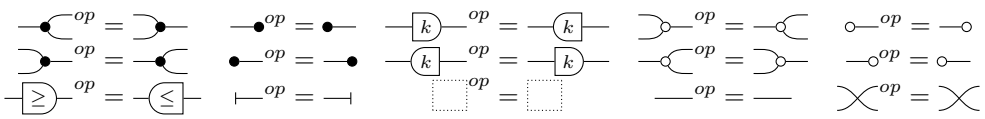


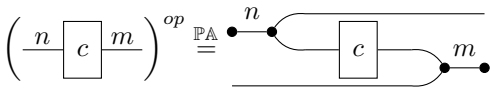
Figure 2 Axioms of $\mathbb{PA}_{\mathsf{k}}$.

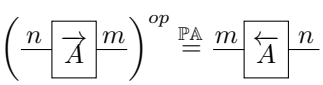
▶ **Definition 9.** *The prop morphism* $\cdot^{op}\colon \mathsf{PDiag}^{op} \to \mathsf{PDiag}$ *is inductively defined as:*



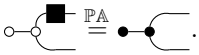
$$(c\,;d)^{op} = d^{op}\,;c^{op} \qquad (c \oplus d)^{op} = c^{op} \oplus d^{op}$$

Observe that $\cdot^{op}$ is controvariant: it maps a diagram $c\colon n \to m$ into $c^{op}\colon m \to n$ which is graphically rendered as the *mirror image* of c: for instance, referring to Example 4, $c^{op} = d$. By exploiting the inductive definition, one can prove that the following hold.


(19)


(20)

if $c \overset{\mathbb{PA}}{\sqsubseteq} d$ then $c^{op} \overset{\mathbb{PA}}{\sqsubseteq} d^{op}$ (21) $\qquad (c^{op})^{op} \overset{\mathbb{PA}}{=} c$ (22)

Equation (19) states that $[\![c^{op}]\!]$ is exactly the *opposite relation* of $[\![c]\!]$, i.e., $[\![c^{op}]\!] = \{(y,x) \in \mathsf{k}^m \times \mathsf{k}^n \mid (x,y) \in [\![c]\!]\}$. In particular, by (20), any diagram in $\mathsf{MD}_{\overrightarrow{\mathsf{iag}}}$ representing a matrix A is mapped into a diagram in $\mathsf{MD}_{\overleftarrow{\mathsf{iag}}}$ representing the same matrix (see Example 4). Thanks to (21) and (22), one has that $c \overset{\mathbb{PA}}{=} d$ iff $c^{op} \overset{\mathbb{PA}}{=} d^{op}$. Therefore, each of the axioms in Figure 2 and each of the laws that we prove in this text can be read both as $c \overset{\mathbb{PA}}{=} d$ and as $c^{op} \overset{\mathbb{PA}}{=} d^{op}$. For instance, by (15) we also know that  . Like in (18), in our derivations we will always use this property implicitly.

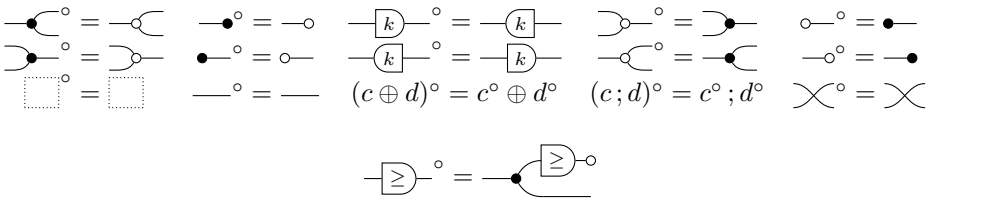
3 The polar operator

When reasoning about cones $C \subseteq \mathsf{k}^n$ in convex algebra, an important role is played by the notions of *polar* and *dual* cone:

$$\mathit{polar}(C) = \{b \in \mathsf{k}^n \mid \forall x \in C,\ b^T x \leq 0\} \qquad \mathit{dual}(C) = \{b \in \mathsf{k}^n \mid \forall x \in C,\ b^T x \geq 0\}$$

As these concepts will also be relevant to our developments, we now study how they are expressible in PCDiag. The fundamental ingredient is the *polar operator* from [5]:

▶ **Definition 10.** *The prop morphism* $\cdot^{\circ}\colon \mathsf{PCDiag} \to \mathsf{PCDiag}$ *is inductively defined as:*

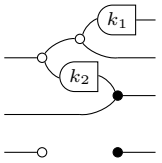


The polar operator subsumes both the concept of dual and polar cone. This can be made precise via the following proposition, whose proof we defer to the end of the next section.

▶ **Proposition 11.** *Let* $C \subseteq \mathsf{k}^n$ *be a polyhedral cone. Let* $c\colon 0 \to n$ *and* $d\colon n \to 0$ *be such that* $[\![c]\!] = \{(\bullet, x) \mid x \in C\}$ *and* $[\![d]\!] = \{(x, \bullet) \mid x \in C\}$*. Then* $[\![c^{\circ}]\!] = \{(\bullet, b) \mid b \in \mathit{polar}(C)\}$ *and* $[\![d^{\circ}]\!] = \{(b, \bullet) \mid b \in \mathit{dual}(C)\}$*.*

Note that Proposition 11 uses two representations of the cone C as a string diagram, one of type $0 \to n$ and the other of type $n \to 0$. Depending on which one we pick, one obtains the polar or the dual of C. Another interesting departure from the traditional approaches is that the polar/dual cone is now specified *inductively* on the structure of the string diagram, following Definition 10. We now provide some properties and examples of the polar operator. First we observe how it behaves on string diagrams representing matrices.

▶ **Example 12.** Consider the matrix A in Example 4 and its encoding as the diagram $c\colon 3 \to 4$ in $\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$. Applying the polar operator on c yields the following diagram in $\mathsf{MD}^{\leftarrow}_{\mathsf{iag}}$

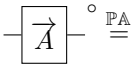
$c^\circ =$ 

$$A^T = \begin{pmatrix} k_1 & 1 & k_2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

representing the transpose A^T of the matrix A. Indeed, $[\![c^\circ]\!] = \{(x, y) \in \mathsf{k}^3 \times \mathsf{k}^4 \mid A^T y = x\}$. This is an instance of a more general phenomenon: when applied to matrices (i.e. string diagrams of $\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$), the polar operator yields their transpose matrix, represented by a string diagram in $\mathsf{MD}^{\leftarrow}_{\mathsf{iag}}$ (and thus to be read "right-to-left').

▶ **Lemma 13** (From [36]). *For all* 

$: n \to m$ *in* $\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$*, the following holds*



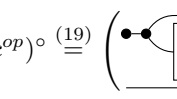
▶ **Proposition 14** (From [5]). *For all diagrams* $c, d\colon n \to m$ *in* PCDiag*, the following hold*

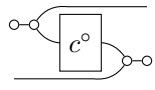
1. *if* $c \overset{\mathbb{PA}}{\sqsubseteq} d$ *then* $(d)^\circ \overset{\mathbb{PA}}{\sqsubseteq} (c)^\circ$*;*
2. $(c^\circ)^\circ \overset{\mathbb{PA}}{=} c$*.*

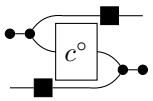
The first item of the above proposition informs us that if $c \overset{\mathbb{PA}}{=} d$ then one can safely conclude that $c^\circ \overset{\mathbb{PA}}{=} d^\circ$. Viceversa, if $c^\circ \overset{\mathbb{PA}}{=} d^\circ$, by the second item, $c \overset{\mathbb{PA}}{=} d$. The next lemma illustrates the interaction of the polar operator with $\cdot^{op}$ (see Definition 9).

▶ **Lemma 15.** *For all* $c\colon n \to m$ *in* PCDiag*, the following holds* $(c^{op})^\circ \overset{\mathbb{PA}}{=} m \blacksquare^{m}\,;(c^\circ)^{op}\,;{}^{n}\blacksquare^{n}$.

Proof.

$(c^{op})^\circ \overset{(19)}{=}$ 

$^\circ =$ 

$\overset{(15)}{=}$ 

$\overset{(19)}{=} \blacksquare\,;(c^\circ)^{op}\,;\blacksquare$ ◀

▶ **Example 16.** The diagrams $\circ\!-\!(\leq)\!-$: $0 \to 1$ and $-\!(\geq)\!-\circ$: $1 \to 0$, denoting the relations $\{(\bullet, x) \mid x \geq 0\} \subseteq \mathsf{k}^0 \times \mathsf{k}^1$ and $\{(x, \bullet) \mid x \geq 0\} \subseteq \mathsf{k}^1 \times \mathsf{k}^0$, are two different representations for the same object in traditional algebra: the polyhedral cone $\{x \in \mathsf{k} \mid x \geq 0\} \subseteq \mathsf{k}^1$. Interestingly enough, applying the polar operator to them yields two different results. Analogous considerations hold for $\{x \in \mathsf{k} \mid x \leq 0\} \subseteq \mathsf{k}^1$.

$(-\!(\geq)\!-\circ)^\circ = -\!(\geq)\!-^\circ\,;-\circ^\circ = -\!\bullet\!(\geq)\!-\circ\,;-\!\bullet \overset{counl}{=} -\!(\geq)\!-\circ$ (23)

$(\circ\!-\!(\leq)\!-)^\circ = (-\!(\geq)\!-\circ^{op})^\circ \overset{Lemma\ 15}{=} (-\!(\geq)\!-\circ^\circ)^{op}\,;\blacksquare \overset{(23)}{=} -\!(\geq)\!-\circ^{op}\,;\blacksquare \overset{\substack{P6\\ zer}}{=} \circ\!-\!(\geq)\!-$ (24)

$(\circ\!-\!(\geq)\!-)^\circ \overset{(24)}{=} (\circ\!-\!(\leq)\!-^\circ)^\circ \overset{Prop.\ 14.2}{=} \circ\!-\!(\leq)\!-$ (25)

$(-\!(\leq)\!-\circ)^\circ = (\circ\!-\!(\geq)\!-^{op})^\circ \overset{Lemma\ 15}{=} \blacksquare\,;(\circ\!-\!(\geq)\!-^\circ)^{op} \overset{(25)}{=} \blacksquare\,;\circ\!-\!(\leq)\!-^{op} \overset{\substack{P6\\ zer}}{=} -\!(\leq)\!-\circ$ (26)

Observe that for the two diagrams above of type $1 \to 0$, $\cdot^\circ$ act as identity, while for those of type $0 \to 1$, it reverses the sign. This behaviour is justified by Proposition 11.

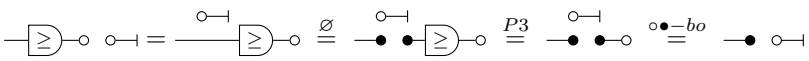
There is a number of other observations about the polar operator, which may be proven with graphical reasoning taking advantage of the inductive definition, the complete axiomatisation and the laws illustrated so far. While this material is not essential to our developments, we conclude this section with two simple "exercises" of that kind, which are left to the interested reader.

- **(Exercise 1)** Prove that, for all c in the form of (8), there exists some d in the form of (9) such that $c^\circ \overset{\mathbb{PA}}{=} d$. *Hint:* use Lemma 13 and *cc–1*
- **(Exercise 2)** Prove that, for all c in the form of (10), there exists some d in the form of (11) such that $c^\circ \overset{\mathbb{PA}}{=} d$. *Hint:* use (23).

4 Lemma of the alternatives

This section is devoted to the diagrammatic formulation of a lemma of *alternatives*, asserting that exactly one of two systems of linear inequalities (i.e. polyhedra) has a solution.

To approach the lemma, an important question is how to model "does a system have a solution?" in our graphical calculus. We focus attention on two morphisms of PDiag of type $0 \to 0$: the empty diagram and the diagram $\circ\!\!-\!\!\dashv$. Intuitively, $\circ\!\!-\!\!\dashv$ asserts that "0 = 1"; its denotational semantics is the composition of the relations $\{(\bullet, 0)\}$ and $\{(1, \bullet)\}$, which gives the empty relation $\emptyset$. Since for any relation R in Rel_k, $R \oplus \emptyset = \emptyset = \emptyset \oplus R$, the behaviour of $\circ\!\!-\!\!\dashv$ resembles that of a logical false. From the viewpoint of the equational theory, $\circ\!\!-\!\!\dashv$ introduces an inconsistency; in particular, by means of the axiom $\varnothing$ we are able to prove that $\circ\!\!-\!\!\dashv \oplus c \overset{\mathbb{PA}}{=} \circ\!\!-\!\!\dashv \oplus d$ for any $c, d \colon n \to m$ in PDiag. As an example, consider the following equation:



(27)

In an analogous way, the behaviour of the diagram can be regarded as a logical true. In particular, its semantics is the relation $id_0 = \{(\bullet, \bullet)\}$ which for any R in Rel_k is such that $R \oplus id_0 = R = id_0 \oplus R$.

Finally, note that in Rel_k the only possible morphisms of type $0 \to 0$ are exactly $\emptyset$ and id_0. Thus the following lemma holds.

▶ **Lemma 17** (From [5]). *For any diagram $c \colon 0 \to 0$ of* PDiag*, either $c \overset{\mathbb{PA}}{=}$  *or $c \overset{\mathbb{PA}}{=} \circ\!\!-\!\!\dashv$*

▶ **Lemma 18** (Lemma of the alternatives). *Let $c \colon 0 \to 1$ be a diagram in* PCDiag*. Then exactly one of the following two equations holds:*

(a) $c \,;\, \dashv \overset{\mathbb{PA}}{=}$  *(b)* $c^\circ \,;\, \dashv \overset{\mathbb{PA}}{=}$  .

Proof. Since c is in PCDiag, then $[\![c]\!] \subseteq \mathsf{k}^0 \times \mathsf{k}^1$ is a polyhedral cone. Thus $[\![c]\!]$ must be one of the following:

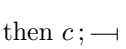
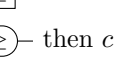
$$\{(\bullet, k) \mid k \in \mathsf{k}\} \quad \{(\bullet, k) \mid k \geq 0\} \quad \{(\bullet, k) \mid k \leq 0\} \quad \{(\bullet, k) \mid k = 0\}$$

By Theorem 8, it holds[1] that either

$c \overset{\mathbb{PA}}{=} \bullet\!\!-$ or $c \overset{\mathbb{PA}}{=} \circ\!\!-\!\!\boxed{\leq}\!\!-$ or $c \overset{\mathbb{PA}}{=} \circ\!\!-\!\!\boxed{\geq}\!\!-$ or $c \overset{\mathbb{PA}}{=} \circ\!\!-$.

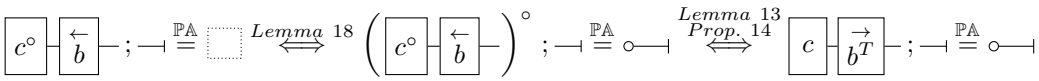
[1] See Appendix A for a purely equational proof that does not invoke completeness.

By Proposition 14.1, we can thus consider only these four cases:

- If $c \overset{\mathbb{PA}}{=}$  then $c\,;\dashv \overset{del}{=}$  and $c^\circ\,;\dashv \overset{Prop.\ 14}{=}$  $= \circ\!\!-\,;\dashv = \circ\!\!-\!\!\dashv$.
- If $c \overset{\mathbb{PA}}{=}$  then $c\,;\dashv \overset{AP1}{=}$  and $c^\circ\,;\dashv \overset{Prop.\ 14}{=}$  $\overset{(24)}{=}$  $\overset{(14)}{=} \circ\!\!-\!\!\dashv$.
- If $c \overset{\mathbb{PA}}{=}$  then $c\,;\dashv \overset{(14)}{=} \circ\!\!-\!\!\dashv$ and $c^\circ\,;\dashv \overset{Prop.\ 14}{=}$ $\overset{(25)}{=}$ $\overset{AP1}{=}$  .
- If $c \overset{\mathbb{PA}}{=} \circ\!\!-$ then $c\,;\dashv \overset{\mathbb{PA}}{=} \circ\!\!-\!\!\dashv$ and $c^\circ\,;\dashv \overset{Prop.\ 14}{=}$  $= \bullet\!\!-\,;\dashv \overset{del}{=}$. ◀

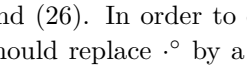
The lemma of alternatives yields as a corollary a proof of Proposition 11.

Proof of Proposition 11. Observe that



By definition of $[\![\cdot]\!]$, the former equation holds iff $(\bullet, b) \in [\![c^\circ]\!]$, while the latter holds iff $\forall (\bullet, x) \in [\![c]\!]$, $b^T x \neq 1$. That is $[\![c^\circ]\!]$ is the relation $\{(\bullet, b) \mid \forall x \in C,\ b^T x \neq 1\}$ which is readily seen to be equal to $\{(\bullet, b) \mid b \in polar(C)\}$.

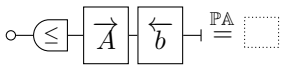
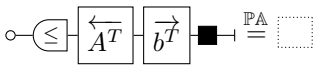
For d, note that $d \overset{\mathbb{PA}}{=} c^{op}$. Thus, by Lemma 15, $d^\circ \overset{\mathbb{PA}}{=}$  $\,;(c^\circ)^{op}$. Thus $(b, \bullet) \in [\![d^\circ]\!]$ iff $(\bullet, -b) \in [\![c^\circ]\!]$ iff $-b \in polar(C)$ iff $b \in dual(C)$. That is $[\![d^\circ]\!] = \{(b, \bullet) \mid b \in dual(C)\}$. ◀

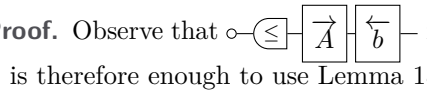
▶ Remark 19. Interestingly, the lemma of alternatives does not hold for diagrams $c\colon 1 \to 0$ (when taking $\vdash\,; c$ and $\vdash\,; c^\circ$ in place of $c\,;\dashv$ and $c^\circ\,;\dashv$): it is easy to see this with (23) and (26). In order to obtain a lemma of alternatives for diagrams of type $c\colon 1 \to 0$, one should replace $\cdot^\circ$ by a novel operator $\cdot^\bullet$ defined as  and as $c^\bullet = c^\circ$ for all the other generators c. Such operator behaves as the dual for diagrams $c\colon 0 \to n$ and as the polar for diagrams $d\colon n \to 0$.

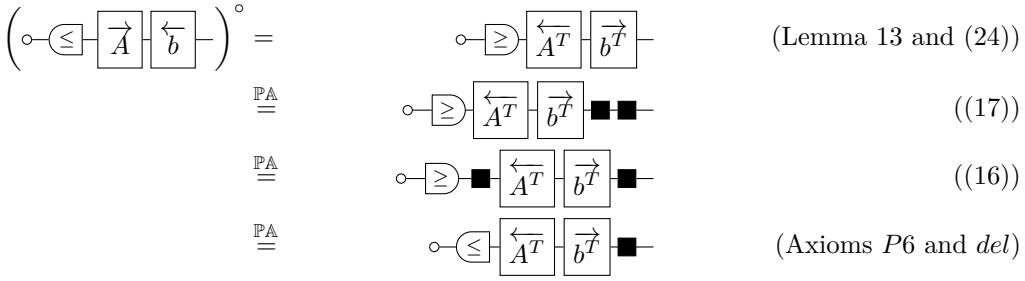
5 A string diagrammatic proof of Farkas' Lemma

The lemma of alternatives provides a direct route to a diagrammatic proof of Farkas' lemma.

▶ **Lemma 20** (Farkas' lemma). *Let* $\overrightarrow{A}\colon n \to m$ *be a diagram in* $\mathsf{MD}^{\rightarrow}_{\mathsf{iag}}$ *and* $\overleftarrow{b}\colon m \to 1$ *in* $\mathsf{MD}^{\leftarrow}_{\mathsf{iag}}$*, then exactly one of the following two equations holds:*

(a)  *(b)* 

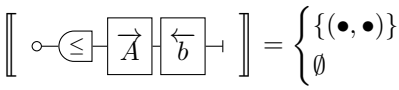
Proof. Observe that  is a diagram $c\colon 0 \to 1$ in PCDiag. In order to conclude, it is therefore enough to use Lemma 18 and observe that



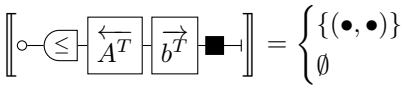
(Lemma 13 and (24)); ((17)); ((16)); (Axioms $P6$ and del)

◀

It is instructive to make explicit in which sense Lemma 20 amounts to the well known result of Farkas. By using the inductive definition of $[\![\cdot]\!]$, one may compute the semantics on the left hand sides of the equations (a) and (b):



$$= \begin{cases} \{(\bullet,\bullet)\} & \text{if } \exists x \in \mathsf{k}^n \text{ s.t. } x \geq 0 \text{ and } Ax = b \\ \emptyset & \text{otherwise} \end{cases}$$



$$= \begin{cases} \{(\bullet,\bullet)\} & \text{if } \exists y \in \mathsf{k}^m \text{ s.t. } A^T y \geq 0 \text{ and } b^T y = -1 \\ \emptyset & \text{otherwise} \end{cases}$$

Therefore equation (a) holds if and only if $\exists x \in \mathsf{k}^n$ s.t. $x \geq 0$ and $Ax = b$ while equation (b) if and only if $\exists y \in \mathsf{k}^m$ s.t. $A^T y \geq 0$ and $b^T y = -1$. In the usual presentation of the Farkas' lemma, e.g. [20], the former condition is exactly the same, while the second one is often expressed by the equivalent condition $\exists y \in \mathsf{k}^m$ s.t. $A^T y \geq 0$ and $b^T y < 0$.[2]

6 Duality in linear programming

Farkas' lemma is closely related to *linear programming*, as it is one of the main tools to prove duality results in this area. In this section, we explore such duality theorems in the context of diagrammatic polyhedral algebra; it turns out that our formulation does not require the direct application of Farkas' lemma, but rather relies on more general principle (Theorem 23) which allows to prove all the results at once.

Duality in linear programming studies pairs of problems of the following shape

$$(P) \coloneqq \quad max\{cx \mid Ax \leq b, x \geq 0\} \qquad\qquad (D) \coloneqq \quad min\{b^T y \mid A^T y \geq c^T \text{ and } y \geq 0\}$$

where A, b and c are matrices of type $m \times n$, $1 \times m$ and $n \times 1$, respectively. The *primal problem* (P) requires to maximise cx subject to the condition that $Ax \leq b$ and $x \geq 0$. Its *dual problem* (D) requires to minimise $b^T y$ subject to the condition that $A^T y \geq c^T$ and $y \geq 0$. The farmer problem and the one of the board from the Introduction are instances of (P) and (D). The primal problem has three possible outcomes: (P) may be *unfeasible*, in the sense that there exists no $x \geq 0$ such that $Ax \leq b$; it can be *unbounded*, when the latter inequality holds for some non-negative vectors $x \in \mathsf{k}^n$, but there exists no maximum $k \in \mathsf{k}$ for cx; or it can be *bounded*, if such k exists. The same possibilities apply to (D).

Duality theory in linear programming establishes a series of possibilities between these possible outcomes: in particular, if (P) is unbounded then (D) is unfeasible and, viceversa, if (D) is unbounded then (P) is unfeasible. Moreover, (P) is bounded if and only if (D) is bounded. The following table summarises such results.

(P) \ (D)	bounded	unbounded	unfeasible
bounded	✓		
unbounded			✓
unfeasible		✓	✓

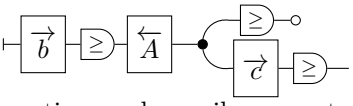
(28)

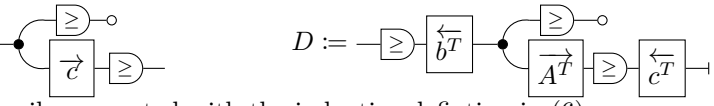
[2] By mean of Proposition 11 one can also translate our proof in traditional algebraic language: first observe that for all one-dimensional polyhedral cones C either 1 belongs to C or 1 belong to the polar of C (Lemma 18); then prove that the polar of $\{z \in \mathsf{k} \mid \exists x \in \mathsf{k}^n \text{ s.t. } x \geq 0 \text{ and } Ax = bz\}$ is exactly $\{z \in \mathsf{k} \mid \exists y \in \mathsf{k}^m \text{ s.t. } A^T y \geq 0 \text{ and } b^T y = -z\}$ (proof of Lemma 20). We could not find the same proof in literature, but it is hard to claim that it does not exist.

The most useful fact is that when (P) and (D) are bounded, they have the same result, i.e.,

$$max\{cx \mid Ax \leq b, x \geq 0\} = min\{b^T y \mid A^T y \geq c^T \text{ and } y \geq 0\}. \tag{29}$$

We now turn to the question of modelling the primal problem (P) and the dual problem (D) in PDiag. Let us fix $\overleftarrow{A} : m \to n$ in $\mathsf{MD}^{\overleftarrow{\mathsf{iag}}}$, $\overrightarrow{b} : 1 \to m$ and $\overrightarrow{c} : n \to 1$ in $\mathsf{MD}^{\overrightarrow{\mathsf{iag}}}$, and consider the following diagrams in PDiag

$P :=$ 

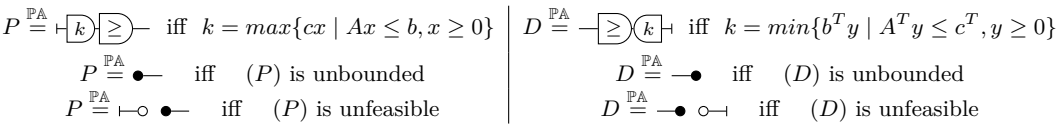
$D :=$ 

Their semantics can be easily computed with the inductive defintion in (6):

$$[\![P]\!] = \{(\bullet, z) \in \mathsf{k}^0 \times \mathsf{k}^1 \mid z \leq cx, x \geq 0, Ax \leq b\}$$

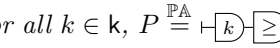
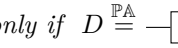
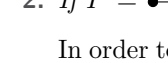
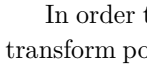
$$[\![D]\!] = \{(z, \bullet) \in \mathsf{k}^1 \times \mathsf{k}^0 \mid z \geq b^T y, y \geq 0, A^T y \geq c^T\}$$

As expected, P models the primal problem (P) and D its dual (D). Indeed, (P) is bounded if and only if $[\![P]\!] = \{(\bullet, z) \mid z \leq k\}$, where k is exactly $max\{cx \mid Ax \leq b, x \geq 0\}$. Also, (P) is unbounded if and only if $[\![P]\!] = \{(\bullet, z) \mid z \in \mathsf{k}\}$ and (P) is unfeasible if and only if $[\![P]\!] = \emptyset$. Analogous considerations hold for (D). The three possibilities can then be expressed in equational terms as follows.



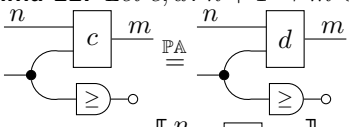
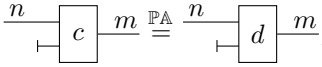
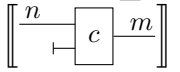
In light of this analysis, the results in (28) and (29) amount to the following theorem.

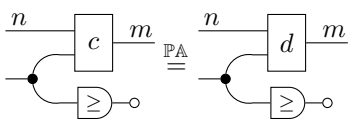
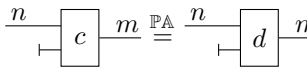
▶ **Theorem 21** (Duality). *The following hold:*

1. *For all* $k \in \mathsf{k}$, $P \overset{\mathbb{PA}}{=}$  *if and only if* $D \overset{\mathbb{PA}}{=}$ 
2. *If* $P \overset{\mathbb{PA}}{=}$  , *then* $D \overset{\mathbb{PA}}{=}$ 
3. *If* $D \overset{\mathbb{PA}}{=}$, *then* $P \overset{\mathbb{PA}}{=}$

In order to prove the above theorem, we exploit homogenisation, a traditional technique to transform polyhedra into cones. The *homogenisation* of polyhedron $P = \{x \in \mathsf{k}^n \mid Ax + b \geq 0\}$ is the polyhedral cone $P^H = \{(x, y) \in \mathsf{k}^{n+1} \mid Ax + by \geq 0, y \geq 0\}$. It holds that $P_1^H = P_2^H$ if and only if $P_1 = P_2$ for all *non-empty* polyhedra P_1, P_2 (see e.g. Lemma 22 in [5]). By exploiting the normal forms in (12) and (10), one obtains the following useful lemma.

▶ **Lemma 22.** *Let* $c, d : n + 1 \to m$ *be string diagrams in* PCDiag.

1. *If*  *then*  ;
2. *Moreover when*  $\neq \emptyset$, *the other implication holds, that is:*

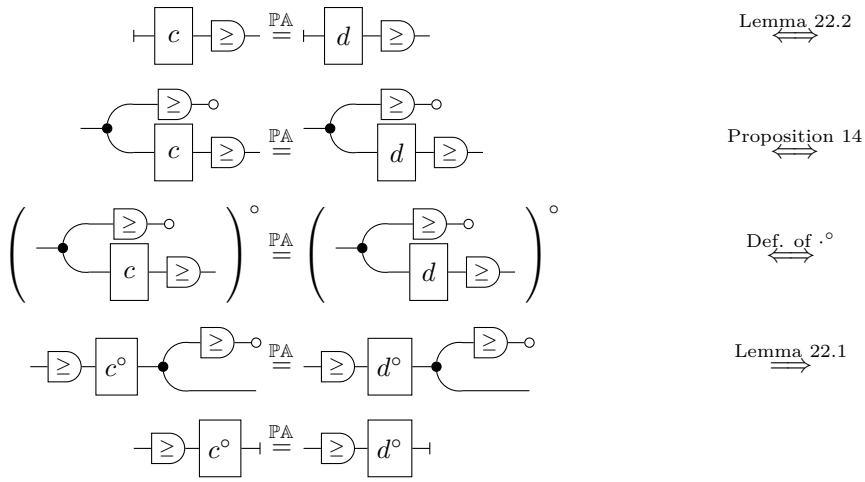
 *iff*  .

By combining homogenisation with the polar operator, we obtain a general proof schema which includes as a particular cases the three points of Theorem 21.

▶ **Theorem 23.** *Let* $c, d\colon 1 \to 1$ *be diagrams in* PCDiag.

1. *If* $[\![c]\!] \neq \emptyset$, *then* $c \overset{\mathbb{PA}}{=} d \implies c^\circ \overset{\mathbb{PA}}{=} d^\circ$;
2. *If* $[\![c^\circ]\!] \neq \emptyset$, *then* $c \overset{\mathbb{PA}}{=} d \impliedby c^\circ \overset{\mathbb{PA}}{=} d^\circ$.

Proof. For the first statement, observe that

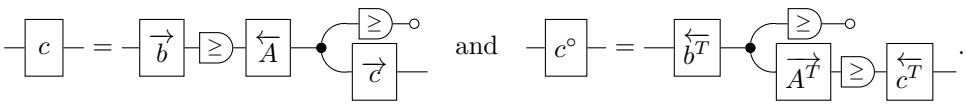


The first step uses Lemma 22.2 because, by hypothesis, the two diagrams denote a non-empty relation. Also, note that the last step uses only the first item of Lemma 22 –thus it is only an implication– because we do not know whether the string diagram denote the empty relation.

To prove the second statement, we use the derivation above, but in the first step we replace $\iff$ by $\impliedby$ (Lemma 22.1) and in the last step we replace $\implies$ by $\iff$ (Lemma 22.2). ◀

From Theorem 23 one may immediately derive the three dualities in Theorem 21.

Proof of Theorem 21. First observe that $P =$  and $D =$ where

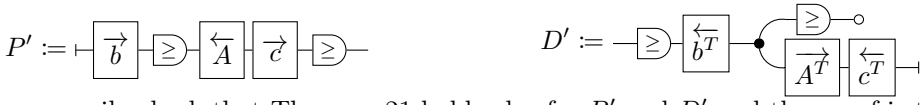


1. Since $[\![k \geq]\!] \neq \emptyset$ and $[\![\geq k]\!] \neq \emptyset$, then one can exploit the two implications of Theorem 23, by taking $d = k$ and observe that $(k)^\circ = k$.
2. Since $[\![\bullet]\!] \neq \emptyset$, then one can use Theorem 23.1 with $d = \bullet\,\bullet$ and observe that $\bullet\,\bullet \geq \overset{del}{\underset{P3}{=}} \bullet$ and $\geq ; (\bullet\,\bullet)^\circ ; \dashv = \geq \circ\!\dashv \overset{(27)}{=} \bullet \circ\!\dashv$.
3. Since $[\![\bullet]\!] \neq \emptyset$, then use Theorem 23.2 with $d^\circ = \bullet\,\bullet$ and proceed as in 2. ◀

▶ Remark 24. Traditional textbooks do not prove duality results for problems in the form of (P) and (D) above, but they need to first massage problems to obtain the following shape.

$$(P') \coloneqq \quad max\{cx \mid Ax \leq b\} \qquad\qquad (D') \coloneqq \quad min\{b^T y \mid A^T y = c^T \text{ and } y \geq 0\}$$

Thanks to Theorem 23, we do not really need to rely on a specific form. Indeed by taking



one can easily check that Theorem 21 holds also for P' and D' and the proof is the same, modulo the obvious choice of c.

7 Conclusions

This paper investigates Farkas' lemma and duality in linear programming within the language of diagrammatic polyhedral algebra. Besides the elegance of the proofs, the linguistic aspect is, in our opinion, the most interesting angle. Indeed, this work can be thought as an exercise in diagrammatic algebra, illustrating its appeal in the following ways:

- by identifying the right primitive components and the appropriate ways to compose them, one is able to express exactly all the objects of interest (in this case, polyhedra) and to formally reason about them by means of a sound and complete axiomatisation ($\mathbb{PA}$);
- operations on few primitives can be extended inductively to all the objects of interest, resulting in an effective way to compute sophisticated notions, like polar and dual cones;
- equations amongst diagrams can express complex statements, like those about the existence of a solution, the maximal or the minimal solution;
- symbolic manipulation of diagrams by means of axioms and derived laws allows to prove such statements.

The last point leads us to believe that our proofs are suitable to be formalised in proof assistants, such as Coq or Agda. Finally, we think that this work may inspire further duality results in string diagrammatic languages other than diagrammatic polyhedral algebra.

References

1 David Avis and Bohdan Kaluzny. Solving inequalities and proving Farkas's lemma made easy. *The American Mathematical Monthly*, 111(2):152–157, 2004.

2 John C Baez and Jason Erbele. Categories in control. *Theory and Applications of Categories*, 30(24):836–881, 2015.

3 David Bartl. A short algebraic proof of the Farkas lemma. *SIAM Journal on Optimization*, 19(1):234–239, 2008.

4 Dimitri P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

5 Guillaume Boisseau, Filippo Bonchi, Alessandro Di Giorgio, and Pawel Sobocinski. Diagrammatic polyhedral algebra, 2021. `arXiv:2105.10946`.

6 Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. *Proceedings of the 46th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, 3:1–28, 2019.

7 Filippo Bonchi, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Graphical affine algebra. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2019.

8 Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In *Proceedings of the 25th International Conference on Concurrency Theory (CONCUR)*, pages 435–450. Springer, 2014.

9 Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. The calculus of signal flow diagrams I: linear relations on streams. *Information and Computation*, 252:2–29, 2017.

10 CG Broyden. A simple algebraic proof of Farkas's lemma and related theorems. *Optimization methods and software*, 8(3-4):185–199, 1998.

11 Roberto Bruni, Ivan Lanese, and Ugo Montanari. A basic algebra of stateless connectors. *Theoretical Computer Science*, 366(1–2):98–120, 2006.

12 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, 2011.

13 Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes - A first course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

14 René David and Hassane Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Berlin, 2 edition, 2010. doi:10.1007/978-3-642-10669-9.

15 Achiya Dax. An elementary proof of Farkas' lemma. *SIAM Review*, 39(3):503–507, 1997. URL: http://www.jstor.org/stable/2133044.

16 Gyula Farkas. A Fourier-féle mechanikai elv alkalmazásának algebrai alapja [hungarian; on the algebraic foundation of the applications of the mechanical principle of Fourier]. *Mathematikai és Physikai Lapok*, 5:49–54, 1896.

17 J. Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902:1–27, 1902.

18 Brendan Fong, Paolo Rapisarda, and Paweł Sobociński. A categorical approach to open and interconnected dynamical systems. In *LICS 2016*, 2016.

19 Brendan Fong and David I. Spivak. String diagrams for regular logic (extended abstract). In John Baez and Bob Coecke, editors, *Proceedings Applied Category Theory 2019, ACT 2019, University of Oxford, UK, 15-19 July 2019*, volume 323 of *EPTCS*, pages 196–229, 2019. doi:10.4204/EPTCS.323.14.

20 David Gale, Harold W Kuhn, and Albert W Tucker. Linear programming and the theory of games. *Activity analysis of production and allocation*, 13:317–335, 1951.

21 Dan R Ghica and Achim Jung. Categorical semantics of digital circuits. In *Proceedings of the 16th Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 41–48, 2016.

22 RA Good. Systems of linear relations. *SIAM Review*, 1(1):1–31, 1959.

23 Alexandre Goy and Daniela Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 454–464. ACM, 2020. doi:10.1145/3373718.3394795.

24 Murray C Kemp and Yoshio Kimura. Introduction to mathematical economics. Technical report, Springer, 1978.

25 Vilmos Komornik. A simple proof of Farkas' lemma. *The American Mathematical Monthly*, 105(10):949–950, 1998. doi:10.1080/00029890.1998.12004992.

26 Stephen Lack. Composing PROPs. *Theory and Application of Categories*, 13(9):147–163, 2004.

27 Saunders Mac Lane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71:40–106, 1965.

28 Olvi L Mangasarian. *Nonlinear programming*. SIAM, 1994.

29 Koko Muroya, Steven W. T. Cheung, and Dan R. Ghica. The geometry of computation-graph abstraction. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 749–758. ACM, 2018. doi:10.1145/3209108.3209127.

30 Hukukane Nikaido. *Convex structures and economic theory*. Elsevier, 2016.

31 Martin H. Pearl. A matrix proof of Farkas's Theorem. *The Quarterly Journal of Mathematics*, 18(1):193–197, January 1967. doi:10.1093/qmath/18.1.193.

32 Robin Piedeleu and Fabio Zanasi. A string diagrammatic axiomatisation of finite-state automata. In *FoSSaCS 2021*, 2021.

33 Peter Selinger. A survey of graphical languages for monoidal categories. *Springer Lecture Notes in Physics*, 13(813):289–355, 2011.

34 Claude E. Shannon. The theory and design of linear differential equation machines. Technical report, National Defence Research Council, 1942.

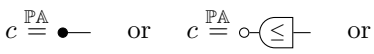
35 Robert J Vanderbei et al. *Linear programming*, volume 3. Springer, 2015.

36 Fabio Zanasi. *Interacting Hopf Algebras: the theory of linear systems*. PhD thesis, Ecole Normale Supérieure de Lyon, 2015.

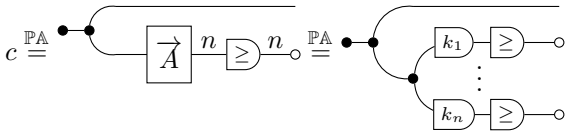
37 Fabio Zanasi. The algebra of partial equivalence relations. *Electronic Notes in Theoretical Computer Science*, 325:313–333, 2016.

A Proofs of Section 4

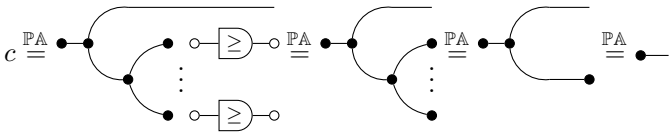
Alternative proof of Lemma 18. We prove diagrammatically, without relying on Theorem 8, that the following property holds for any $c\colon 0 \to 1$ in PCDiag



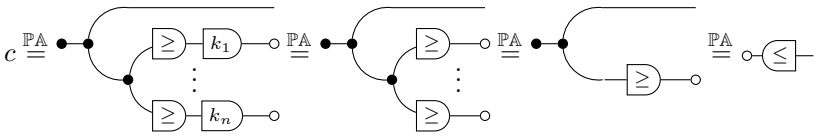
Any $n \to m$ diagram in PCDiag is equivalent to one in the form of (10). A diagram $c\colon 0 \to 1$ has the following normal form, where A is a $n \times 1$ matrix



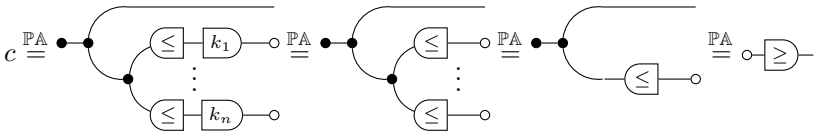
- If $k_i = 0$ for all $i = 1 \dots n$, then



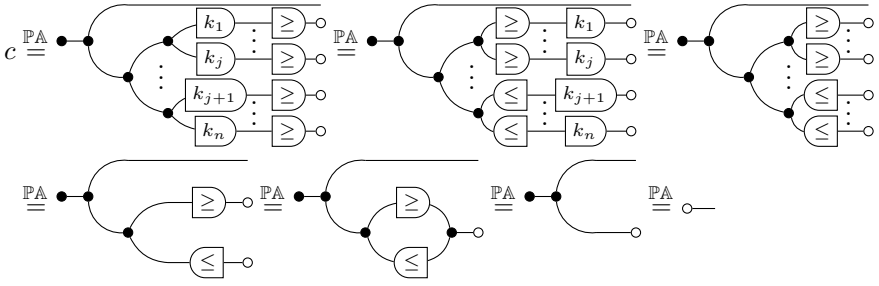
- If $k_i \geq 0$ for all $i = 1 \dots n$, then those $k_i = 0$ are just *detached* as in the first case and for the others



- If $k_i \leq 0$ for all $i = 1 \dots n$, then those $k_i = 0$ are just *detached* as in the first case and for the others



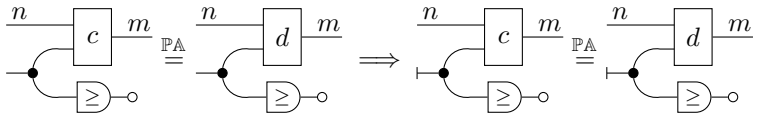
- If some $k_i \geq 0$, and some $k_i \leq 0$, then those $k_i = 0$ are just *detached* as in the first case. For the others, assume without loss of generality that the first j are positive and the remaining ones are negative.



The rest of the proof goes as the original one. ◀

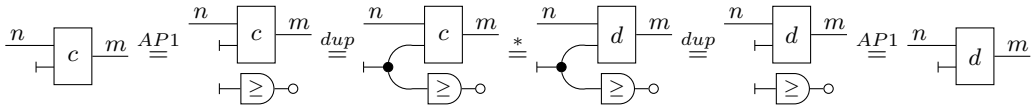
B Proofs of Section 6

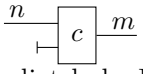
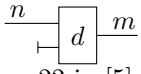
Proof of Lemma 22. To prove 1., first notice that



(*)

Then it is enough to show that



For 2., notice that  and  denote two non-empty polyhedra. Then one can conclude immediately by Lemma 22 in [5] and Theorem 8. ◀

On Doctrines and Cartesian Bicategories

Filippo Bonchi
University of Pisa, Italy

Alessio Santamaria
University of Pisa, Italy

Jens Seeber
University of Pisa, Italy

Paweł Sobociński
Tallinn University of Technology, Estonia

Abstract

We study the relationship between cartesian bicategories and a specialisation of Lawvere's hyperdoctrines, namely elementary existential doctrines. Both provide different ways of abstracting the structural properties of logical systems: the former in algebraic terms based on a string diagrammatic calculus, the latter in universal terms using the fundamental notion of adjoint functor. We prove that these two approaches are related by an adjunction, which can be strengthened to an equivalence by imposing further constraints on doctrines.

2012 ACM Subject Classification Theory of computation → Logic; Theory of computation → Categorical semantics

Keywords and phrases Cartesian bicategories, elementary existential doctrines, string diagram

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.10

Related Version *Extended Version*: https://arxiv.org/abs/2106.08142

Funding *Filippo Bonchi, Alessio Santamaria*: Supported by the Ministero dell'Università e della Ricerca of Italy under Grant No. 201784YSZ5, PRIN2017 – ASPRA.
Paweł Sobociński: Supported by the ESF funded Estonian IT Academy research measure (project 2014-2020.4.05.19-0001) and the Estonian Research Council grant PRG1210.

Acknowledgements We thank Pino Rosolini and Fabio Pasquali for discussions that improved our understanding of doctrines.

1 Introduction

In [21, 22, 23] Lawvere introduced the notion of *hyperdoctrine* in an effort to capture the *universal* content of logical theories, and first-order logic in particular. Here, by universal, we intend by means of *universal properties* in category theory. The starting point is the notion of Lawvere theory [20], the universal way of capturing the notion of algebraic theory – where the universal property is that of cartesian categories, namely categories with finite products. In terms of logical content, Lawvere theories provide the notion of *term*. Now, a hyperdoctrine is a certain contravariant functor P from the Lawvere theory of terms to a posetal 2-category, e.g. lattices or Heyting algebras. The basic, high-level idea is that the functor takes us from terms to *formulas*; more precisely, the objects of the Lawvere theories, which can be thought of as variable contexts, are taken to the Lindenbaum-Tarski algebra of formulas over these contexts. In this way, the concept of *quantifier* can be captured by means of a universal property – the existence of left-adjoints (existential quantification) and right-adjoints (universal quantification) to the image along P of the projections.

In recent years, there has been a large number [1, 2, 5, 10, 12, 13, 17, 28, 30] of contributions that use string diagrams in order to model computational phenomena of different kinds. Typically, the languages come with an equational theory, which can be used to reason about systems via diagrammatic reasoning. Interestingly, the *same* algebraic structures seem to appear in many different contexts, e.g. commutative monoids and comonoids, Frobenius algebras, Hopf algebras, etc. These applications, while using the language and tools of (monoidal) category theory, are of a rather different nature than the more established "universal" approaches, such as Lawvere theories and hyperdoctrines sketched in the previous paragraph. A bridge between the universal and algebraic worlds is given by a theorem of Fox [15] that characterises cartesian categories as symmetric monoidal categories where each object is equipped with a well-behaved commutative comonoid structure. This means that any Lawvere theory can be seen concretely as a string diagrammatic language (see e.g., [6]). More recently, the notion of discrete cartesian restriction category was characterised in a similar way [11], with partial Frobenius algebras taking the place of commutative comonoids. This raises a natural question: can we capture the universal content of logical theories algebraically in a similar way? In other words, what are the "Fox theorems" for logic?

In this paper we turn our attention to the regular fragment of first-order logic with equality: formulas are built up from terms and the equality relation using the existential quantifier and conjunction. There has been much work on categorifying this fragment, notably the significant corpus of work on allegories [16]. More relevant to our story, we focus on the contrast between universal and algebraic approaches. A universal treatment, the notion of *elementary existential doctrine*, was introduced in [23] and studied extensively in [26]. The basic setup is the same as for Lawvere's hyperdoctrines, but one asks only for the left adjoints, which, as we have previously mentioned, are the universal explanation for existential quantifiers. On the algebraic side, the concept that stands out is that of Carboni and Walters' cartesian bicategories (of relations) [9], which are poset-enriched symmetric monoidal categories where objects are equipped with a special Frobenius algebra and a lax-natural commutative comonoid structures. While Carboni and Walters emphasised the relational algebraic aspects, they were certainly aware of the logical connections. In fact, some recent works [4, 14, 29] exploited various ramifications of the correspondence between cartesian bicategories and regular logic.

Our goal for this paper is a "Fox theorem" for the regular fragment, connecting the universal and the algebraic approaches. Our starting observation is that, given a cartesian bicategory $(\mathbb{B}, \otimes, I)$, one obtains an elementary existential doctrine by restricting the hom functor $\mathrm{Hom}_{\mathbb{B}}(-, I) \colon \mathbb{B}^{\mathrm{op}} \to \mathsf{Set}$ to the (cartesian) category of *maps* of $\mathbb{B}$. In Remark 2.5 of [27], it is mentioned that the other direction is also possible: given an elementary existential doctrine one can construct a cartesian bicategory. We explore the ramifications of this remark in detail. We show that these two translations are functorial and, actually, that they form an adjunction. More precisely, it turns out that the category of cartesian bicategories is a reflective subcategory of the category of doctrines.

The adjunction, however, is *not* an equivalence. We prove this with a counterexample that captures the crux of the matter: there are doctrines $P \colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ where the indexing categories of terms $\mathbb{C}$ are not tailored to the represented logics. In doctrines-as-logical-theories, roughly speaking, equality can come from two places: implicitly, from the indexing term category, and explicitly, via logical equivalence. Doctrines, therefore, have an additional degree of intensionality: doctrines that "substantially represent" the same logic may have distinct index categories and thus not be isomorphic. This issue does not arise in cartesian bicategories where the role of $\mathbb{C}$ is played by the subcategory of maps: maps are arrows satisfying certain properties, rather than given a priori in a fixed index category.

We conclude by observing that, by adding further constraints to the notion of elementary existential doctrine, namely comprehensive diagonals and the Rule of Unique Choice from [27], it is possible to exclude such problematic doctrines. By doing so, we restrict the adjunction to an equivalence, thus obtaining a satisfactory "Fox theorem".

▶ Notation. Given $f\colon X \to Y$ and $g\colon Y \to Z$ morphisms in some category, we denote their composite as $f\,;g$, $g \circ f$ or gf. We write P_f for the action of a doctrine P on a morphism f.

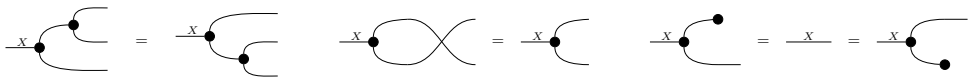
2 Cartesian Categories

The starting point of our exposition is the definition of cartesian category that, thanks to the results of Fox [15], can be given in the following form, which is particularly convenient for the purposes of this paper.

▶ **Definition 1.** *A cartesian category is a symmetric monoidal category $(\mathbb{C}, \otimes, I)$ where every object $X \in \mathbb{C}$ is equipped with morphisms*

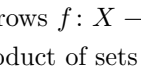
 $: X \to X \otimes X$ *and*  $: X \to I$ *such that*

1.  *and*  *form a cocommutative comonoid, that is they satisfy*



2. *Each morphism $f\colon X \to Y$ is a comonoid homomorphism, that is*

3. *The choice of comonoid on every object is coherent with the monoidal structure in the sense that*

Indeed, given a category $\mathbb{C}$ with finite products, one can construct a monoidal category as in the definition above, by taking as monoidal product $\otimes$ the categorical product and as its unit I the terminal object; for every object X,  is given by the pairing $\langle \mathrm{id}_X, \mathrm{id}_X \rangle\colon X \to X \otimes X$, hereafter denoted by Δ_X, and  by the unique morphism $!_X\colon X \to 1$. Conversely, given a symmetric monoidal category $(\mathbb{C}, \otimes, I)$ as in Definition 1, $\otimes$ forms a categorical product where projections $\pi_X : X \otimes Y \to X$ are given as $id_X \otimes$  and the pairing $\langle f, g \rangle\colon X \to Y \otimes Z$ as  $;(f \otimes g)$ for all arrows $f\colon X \to Y$ and $g\colon X \to Z$.

Hereafter, we will use $\times$ to denote both the cartesian product of sets and the categorical product in an arbitrary cartesian category. It is worth to remark that while Set (sets and functions) and Rel (sets and relations) are both cartesian categories, the categorical product in Set is indeed the cartesian product, while in Rel it is actually the disjoint union.

▶ **Example 2.** Another cartesian category that will play an important role is L_Σ, the *Lawvere theory* [20] generated by a cartesian signature Σ (a set of symbols f equipped with some arity $ar(f) \in \mathbb{N}$). In L_Σ, objects are natural numbers and arrows are tuples of terms over a

$$\frac{i \leq n}{x_i \colon (n,1)}\,(V) \quad \frac{f \in \Sigma \quad ar(f) = m \quad \langle t_1, \dots t_m \rangle \colon (n,m)}{f\langle t_1, \dots, t_m \rangle \colon (n,1)}\,(\Sigma) \quad \frac{t_1 \colon (n,1) \quad \langle t_2, \dots, t_m \rangle \colon (n, m-1)}{\langle t_1, \dots, t_m \rangle \colon (n,m)}\,\langle \dots \rangle \quad \frac{}{\langle\rangle \colon (n,0)}\,\langle\rangle$$

$$\frac{}{\top \colon (n,0)}\,(\top) \quad \frac{P \in \mathbb{P} \quad ar(P) = m \quad \langle t_1, \dots t_m \rangle \colon (n,m)}{P\langle t_1, \dots, t_m \rangle \colon (n,0)}\,(\mathbb{P}) \quad \frac{\phi \colon (n+1,0)}{\exists x_{n+1}.\phi \colon (n,0)}\,(\exists) \quad \frac{\langle t_1, t_2 \rangle \colon (n,2)}{t_1 = t_2 \colon (n,0)}\,(=) \quad \frac{\phi \colon (n,0) \quad \psi \colon (n,0)}{\phi \wedge \psi \colon (n,0)}\,(\wedge)$$

Figure 1 Sort inference rules for L_Σ (top line) and for formulas in regular logic (bottom line).

countable set of variables $V = \{x_1, x_2, \dots\}$. More precisely, arrows from n to m are tuples $\langle t_1, \dots, t_m \rangle$ of sort (n, m) as defined by the inference rules in the first line of Figure 1. It is easy to check that $\langle t_1, \dots, t_m \rangle$ has sort (n, m) if each term t_i has variables in $\{x_1, \dots, x_n\}$. Composition is defined by (simultaneous) substitution: the composition of $\langle t_1, \dots, t_m \rangle \colon (n, m)$ with $\langle s_1, \dots, s_l \rangle \colon (m, l)$ is the tuple $\langle u_1, \dots, u_l \rangle \colon (n, l)$ where $u_i = s_i[^{t_1 \dots t_m}/_{x_1 \dots x_m}]$ for all $i = 1, \dots, l$. One can readily check that L_Σ is a symmetric monoidal category having $(\mathbb{N}, +, 0)$ as the monoid of objects, i.e., it is a *prop* (product and permutation category, see [19, 24]). Identities id_n and symmetries $\sigma_{n,m}$ are defined as expected;  $: n \to n + n$ is the tuple $\langle x_1, \dots, x_n, x_1, \dots, x_n \rangle$ thus acting as a *duplicator* of variables;  $: n \to 0$ is the empty tuple $\langle\rangle$, acting as a *discharger*.

▶ **Definition 3.** *A morphism of cartesian categories is a strict monoidal functor preserving the chosen comonoid structures.*

▶ **Example 4.** Let Σ_1 be the cartesian signature consisting of a single symbol f with arity 1, and Σ_2 be the signature with two symbols, g_1 and g_2, both of arity 1. Consider the corresponding Lawvere theories L_{Σ_1} and L_{Σ_2}. The assignment $f \mapsto g_1$ induces a morphism of cartesian categories, hereafter denoted by $F_1 \colon \mathsf{L}_{\Sigma_1} \to \mathsf{L}_{\Sigma_2}$. Similarly, let $F_2 \colon \mathsf{L}_{\Sigma_1} \to \mathsf{L}_{\Sigma_2}$ denote the morphism of cartesian categories where f is mapped to g_2. Finally, there is a unique morphism of cartesian categories $Q \colon \mathsf{L}_{\Sigma_2} \to \mathsf{L}_{\Sigma_1}$ mapping g_1 and g_2 to f.

3 Elementary Existential Doctrines

Recall that an *inf-semilattice* is a partially ordered set with all finite infima, including a top element $\top$. We denote the category of inf-semilattices and inf-preserving functions by InfSL.

The following definition is taken almost verbatim from [27]. The difference is that there the base category $\mathbb{C}$ only needs binary products, whereas we also require a terminal object.

▶ **Definition 5.** *Let $\mathbb{C}$ be a cartesian category. An elementary existential doctrine is given by a functor $P \colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ that is:*

- Elementary, *namely for every object A in $\mathbb{C}$ there is an element $\delta_A \in P(A \times A)$ such that for every map $e = id_X \times \Delta_A \colon X \times A \to X \times A \times A$, the function $P_e \colon P(X \times A \times A) \to P(X \times A)$ has a left adjoint $\exists_e \colon P(X \times A) \to P(X \times A \times A)$ defined by the assignment*

$$\exists_e(\alpha) = P_{\langle \pi_1, \pi_2 \rangle \colon X \times A \times A \to X \times A}(\alpha) \wedge P_{\langle \pi_2, \pi_3 \rangle \colon X \times A \times A \to A \times A}(\delta_A). \tag{1}$$

- Existential, *namely for every $A_1, A_2 \in \mathbb{C}$ and projection $\pi_i \colon A_1 \times A_2 \to A_i$ with $i \in \{1, 2\}$, the function $P_{\pi_i} \colon P(A_i) \to P(A_1 \times A_2)$ has a left-adjoint $\exists_{\pi_i}$ that satisfies*

- *the* Beck-Chevalley condition*: for any projection* $\pi\colon X \times A \to A$ *and any pullback*

$$\begin{array}{ccc} X' & \xrightarrow{\pi'} & A' \\ {\scriptstyle f'}\downarrow & & \downarrow{\scriptstyle f} \\ X \times A & \xrightarrow{\pi} & A \end{array} \quad \textit{it holds that } \exists_{\pi'}(P_{f'}(\beta)) = P_f(\exists_\pi(\beta)) \textit{ for any } \beta \in P(X \times A). \tag{2}$$

- Frobenius reciprocity*: for any projection* $\pi\colon X \times A \to A$*,* $\alpha \in P(A)$ *and* $\beta \in P(X)$*, it holds that* $\exists_\pi(P_\pi(\alpha) \wedge \beta) = \alpha \wedge \exists_\pi(\beta)$*.*

▶ Remark 6. Taking X in (1) to be the terminal object of $\mathbb{C}$, one obtains that the function $\exists_{\Delta_A}\colon P(A) \to P(A \times A)$ given by

$$\exists_{\Delta_A}(\alpha) = P_{\pi_1}(\alpha) \wedge \delta_A \tag{3}$$

is left-adjoint to $P_{\Delta_A}\colon P(A \times A) \to P(A)$. This condition, which appears in [27], is therefore redundant when $\mathbb{C}$ has terminal object.

▶ Remark 7. In any cartesian category, the diagram below is a pullback of $f\colon A' \to A$ with a projection $\pi\colon X \times A \to A$.

$$\begin{array}{ccc} X \times A' & \xrightarrow{\pi_2} & A' \\ {\scriptstyle \mathrm{id}_X \times f}\downarrow & & \downarrow{\scriptstyle f} \\ X \times A & \xrightarrow{\pi} & A \end{array}$$

Therefore, given a functor $P\colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$, to check that P satisfies the Beck-Chevalley condition it suffices to show that (2) holds for $X' = X \times A'$, $\pi' = \pi_2$ and $f' = \mathrm{id}_X \times f$.

The contravariant powerset $\mathcal{P}$, while often seen as an endofunctor on Set, can also be seen as a functor $\mathcal{P}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{InfSL}$. It is the classical example of an elementary existential doctrine. Recall that, for $f\colon Y \to X$ in Set, $\mathcal{P}_f\colon \mathcal{P}(X) \to \mathcal{P}(Y)$ is $\mathcal{P}_f(Z) = \{y \in Y \mid f(y) \in Z\}$. Given a projection $\pi\colon X \times A \to A$, $\mathcal{P}_\pi$ and its left adjoint $\exists_\pi$ are as follows:

$$\begin{array}{ccc} \mathcal{P}(A) & \xrightarrow{\mathcal{P}_\pi} & \mathcal{P}(X \times A) \\ B & \longmapsto & \{(x,b) \in X \times A \mid b \in B\} \end{array} \qquad \begin{array}{ccc} \mathcal{P}(X \times A) & \xrightarrow{\exists_\pi} & \mathcal{P}(A) \\ S & \longmapsto & \{a \in A \mid \exists x \in X.\, (x,a) \in S\} \end{array}$$

For every set A, δ_A is fixed to be $\{(a,a) \mid a \in A\} \in \mathcal{P}(A \times A)$. With this information, it is easy to check that $\mathcal{P}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{InfSL}$ satisfies the conditions in Definition 5. The reader can find the worked out details in [3].

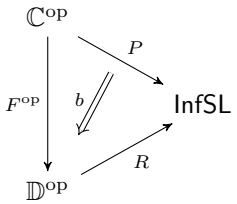
▶ **Example 8.** Let Σ and $\mathbb{P}$ be signatures of function and predicate symbols respectively. The regular fragment of first order logic consists of formulas built from conjunction $\wedge$, true $\top$, existential quantification $\exists$, equality $t_1 = t_2$ and atoms $P\langle t_1, \dots t_m\rangle$ where $P \in \mathbb{P}$ and t_i are terms over Σ. Formulas are sorted according to the rules in Figure 1: ϕ has sort $(n,0)$ if the free variables of ϕ are in $\{x_1, \dots, x_n\}$.

The indexed Lindenbaum-Tarski algebra functor $LT\colon \mathsf{L}_\Sigma^{\mathrm{op}} \to \mathsf{InfSL}$ assigns to each $n \in \mathbb{N}$ the set of formulas of sort $(n,0)$ modulo logical equivalence (defined in the usual way). These form a semilattice with top given by $\top$ and meet by $\wedge$, where $\phi \leq \psi$ if and only if ψ is a logical consequence of ϕ. To the arrow $\langle t_1, \dots, t_m\rangle\colon n \to m$, LT assigns the substitution mapping each $\phi\colon (m,0)$ to $\phi[{}^{t_1,\dots,t_m}/_{x_1,\dots,x_m}]\colon (n,0)$. In particular, for the projection $\pi\colon n+m \to n$ (that is $\langle x_1, \dots, x_n\rangle\colon n+m \to n$) LT_π maps a formula of sort

$(n,0)$ to the same formula but with sort $(n+m,0)$[1]. Its left adjoint $\exists_\pi$ maps a formula $\phi\colon (n+m,0)$ to the formula $\exists x_{n+1}.\ldots\exists x_{n+m}.\,\phi\colon (n,0)$. The Beck-Chevalley condition asserts that "substitution commutes with quantification": if ϕ is a formula with at most $n+1$ free variables and t is a term that does not contain x_{n+1}, then $\exists x_{n+1}.\,(\phi[^t/_{x_i}]) = (\exists x_{n+1}.\,\phi)[^t/_{x_i}]$. Frobenius reciprocity states that $\exists x_i.\,(\phi\wedge\psi) = \phi\wedge(\exists x_i.\,\psi)$ if x_i is not a free variable of ϕ. For all $n\in\mathbb{N}$, δ_n is the formula $(x_1 = x_{n+1})\wedge(x_2 = x_{n+2})\wedge\cdots\wedge(x_n = x_{n+n})$.

▶ **Definition 9** (Cf. [27]). *The category* EED *consists of the following data.*

- *Objects are elementary existential doctrines* $P\colon \mathbb{C}^{\mathrm{op}}\to \mathsf{InfSL}$.
- *Morphisms from* $P\colon \mathbb{C}^{\mathrm{op}}\to\mathsf{InfSL}$ *to* $R\colon \mathbb{D}^{\mathrm{op}}\to\mathsf{InfSL}$ *are pairs* (F,b), *where* $F\colon\mathbb{C}\to\mathbb{D}$ *is a strict cartesian functor while* $b\colon P\to R\circ F^{\mathrm{op}}$ *is a natural transformation*



that preserves equalities and existential quantifiers, that is $b_{A\times A}(\delta^P_A) = \delta^R_{F(A)}$ *for all* A *in* $\mathbb{C}$ *and, for any projection* $\pi\colon X\times A\to A$ *in* $\mathbb{C}$, *the following diagram commutes.*

$$\begin{array}{ccc} P(X\times A) & \xrightarrow{\exists^P_\pi} & P(A) \\ {\scriptstyle b_{X\times A}}\downarrow & & \downarrow{\scriptstyle b_A} \\ RF(X\times A) & \xrightarrow[\exists^R_{F(\pi)}]{} & RF(A) \end{array} \tag{4}$$

- *Composition of* $(F,b)\colon P\to R$ *as above with* $(G,c)\colon R\to S$ *is given by* $(GF, cF\circ b)$.

▶ Remark 10. In [27], morphisms of elementary existential doctrines $P\to R$ are pairs (F,b) where $F\colon\mathbb{C}\to\mathbb{D}$ is a functor between the base categories that preserves binary products merely up to isomorphism, so $F(X)\xleftarrow{F(\pi_1)}F(X\times Y)\xrightarrow{F(\pi_2)}F(Y)$ is a product diagram of $F(X)$ and $F(Y)$ in $\mathbb{D}$ but it might not coincide with the *chosen* product of $\mathbb{D}$. For this reason, preservation of equality for b in [27] means that $b_{A\times A}(\delta^P_A) = R_{\langle F(\pi_1),F(\pi_2)\rangle}(\delta^R_{F(A)})$.

▶ Remark 11. Let $P\colon\mathbb{D}^{\mathrm{op}}\to\mathsf{InfSL}$ be an elementary existential doctrine and $F\colon\mathbb{C}\to\mathbb{D}$ a strict cartesian functor. Then $P\circ F^{\mathrm{op}}\colon\mathbb{C}^{\mathrm{op}}\to\mathsf{InfSL}$ is again an elementary existential doctrine, where $\delta^{PF^{\mathrm{op}}}_A = \delta^P_{F(A)}$ for all A in $\mathbb{B}$ and, for very $\pi_i\colon X_1\times X_2\to X_i$, $\exists^{PF^{\mathrm{op}}}_{\pi_i} = \exists^P_{F(\pi_i)}$.

4 Cartesian Bicategories

In this section we recall from [9] definitions and properties of cartesian bicategories.

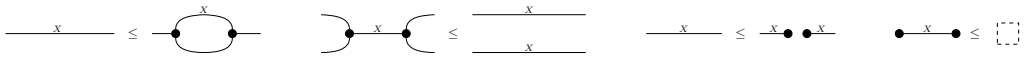
▶ **Definition 12.** *A cartesian bicategory is a symmetric monoidal category* $(\mathbb{B},\otimes,I)$ *enriched over the category of posets (that is every hom-set is a partial order and both composition and* $\otimes$ *are monotonous operations) where every object* $X\in\mathbb{B}$ *is equipped with morphisms*

$: X\to X\otimes X$ *and* 

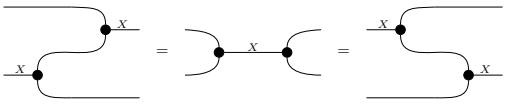
$: X\to I$ *such that*

[1] The second projection $\pi\colon n+m\to m$ requires the reindexing of variables. We do not discuss this case in order to keep the presentation simpler.

1.  *and*  *form a cocommutative comonoid, as in Definition 1.1*
2.  *and*  *have right adjoints*  *and*  *respectively, that is*

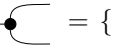


3. *The Frobenius law holds, that is*



4. *Each morphism* $R\colon X \to Y$ *is a* lax *comonoid homomorphism, that is*

5. *The choice of comonoid is coherent with the monoidal structure*[2]*, as in Definition 1.3.*

The archetypal example of a cartesian bicategory is the category of sets and relations Rel, with cartesian product of sets as monoidal product and $1 = \{\bullet\}$ as unit I. To be precise, Rel has sets as objects and relations $R \subseteq X \times Y$ as arrows $X \to Y$. Composition and monoidal product are defined as expected: $R\,;S = \{(x,z) \mid \exists y \text{ s.t. } (x,y) \in R \text{ and } (y,z) \in S\}$ and $R \otimes S = \{\big((x_1,x_2),(y_1,y_2)\big) \mid (x_1,y_1) \in R \text{ and } (x_2,y_2) \in S\}$. For each set X, the comonoid structure is given by the diagonal function $\Delta_X\colon X \to X \times X$ and the unique function $!_X\colon X \to 1$, considered as relations, that is  $= \{\big(x,(x,x)\big) \mid x \in X\}$ and  $= \{(x,\bullet) \mid x \in X\}$. Their right adjoints are given by their opposite relations:  $= \{\big((x,x),x\big) \mid x \in X\}$ and  $= \{(\bullet,x) \mid x \in X\}$. Following the analogy with Rel, we will often call "relations" arbitrary morphisms of a cartesian bicategory.

One of the fundamental properties of cartesian bicategories that follows from the existence of right adjoints (Property 2 in Definition 12) is that every local poset $\mathrm{Hom}_{\mathbb{B}}(X,Y)$ allows to take the intersection of relations and has a top element.

▶ **Lemma 13.** *Let* $\mathbb{B}$ *be a cartesian bicategory and* $X, Y \in \mathbb{B}$*. The poset* $\mathrm{Hom}_{\mathbb{B}}(X,Y)$ *has a top element given by*  *and the meet of relations* $R, S\colon X \to Y$ *is given by*

The Frobenius law (Property 3) gives a compact closed structure – in other words, it allows us to bend wires around. The cup of this compact closed structure is  , the cap analogously  and the Frobenius law implies the snake equations:

 (5)

To obtain an intuition for the lax comonoid homomorphism condition (Property 4), it is useful to spell out its meaning in Rel: in the first inequality, the left and the right-hand side are,

[2] In the original definition of [9] this property is replaced by requiring the uniqueness of the comonoid/-monoid. However, as suggested in [29], coherence seems to be the property of primary interest.

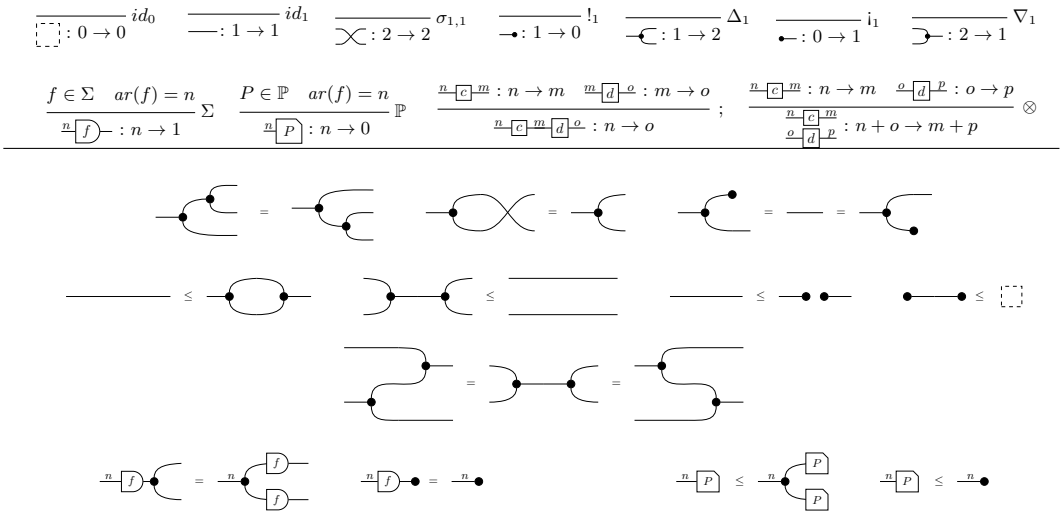
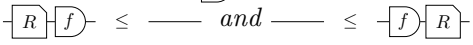
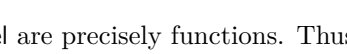


Figure 2 Sort inference rules (top) and axioms (bottom) for $\mathsf{CB}_{\Sigma,\mathbb{P}}$.

respectively, the relations $\{(x,(y,y)) \mid (x,y) \in R\}$ and $\{(x,(y,z)) \mid (x,y) \in R \text{ and } (x,z) \in R\}$, while in the second inequality, they are the relations $\{(x,\bullet) \mid \exists y \in Y \text{ s.t. } (x,y) \in R\}$ and $\{(x,\bullet) \mid x \in X\}$. It is immediate to see that the two left-to-right inclusions hold for any relation $R \subseteq X \times Y$, while the right-to-left inclusions hold for exactly those relations that are (graphs of) functions: the inclusions specify, respectively, single-valuedness and totality.

▶ **Definition 14.** *A map in a cartesian bicategory is an arrow f that is a comonoid homomorphism, i.e. for which the equalities in Definition 1.2 hold. For $\mathbb{B}$ a cartesian bicategory, its category of maps,* $\mathrm{Map}(\mathbb{B})$*, has the same objects of $\mathbb{B}$ and as morphisms the maps of $\mathbb{B}$.*

▶ **Lemma 15.** *A morphism*  *is a map if and only if it has a right adjoint – a morphism R such that*  *and*  .

As expected, maps in Rel are precisely functions. Thus, $\mathrm{Map}(\mathsf{Rel})$ is exactly Set. For maps it makes sense to imagine a flow of information from left to right. We will therefore draw  to denote a map f. Note that we use lower-case letters for maps and upper-case for arbitrary morphisms. Since  and are maps by Lemma 15 and since, by definition, every map respects them, we have that $\mathrm{Map}(\mathbb{B})$, which inherits the monoidal product from $\mathbb{B}$, has the structure of a cartesian category (Definition 1).

▶ **Lemma 16.** *For a cartesian bicategory $\mathbb{B}$, the monoidal product $\otimes$ is a product on* $\mathrm{Map}(\mathbb{B})$*, and the monoidal unit I is terminal. In other words,* $(\mathrm{Map}(\mathbb{B}), \otimes, I)$ *is a cartesian category.*

▶ **Definition 17.** *A morphism of cartesian bicategories is a strict monoidal functor that preserves the ordering, the chosen monoid and the comonoid structures. Cartesian bicategories and their morphisms form a category* CBC.

▶ **Proposition 18.** *Let $F\colon \mathbb{A} \to \mathbb{B}$ be a morphism of cartesian bicategories. Then restricting its domain to* $\mathrm{Map}(\mathbb{A})$ *yields a strict cartesian functor $F{\restriction_{\mathrm{Map}(\mathbb{A})}}\colon \mathrm{Map}(\mathbb{A}) \to \mathrm{Map}(\mathbb{B})$.*

The remaining sections focus on the relationship between cartesian bicategories and elementary existential doctrines. First, a little taste of the similarity between them.

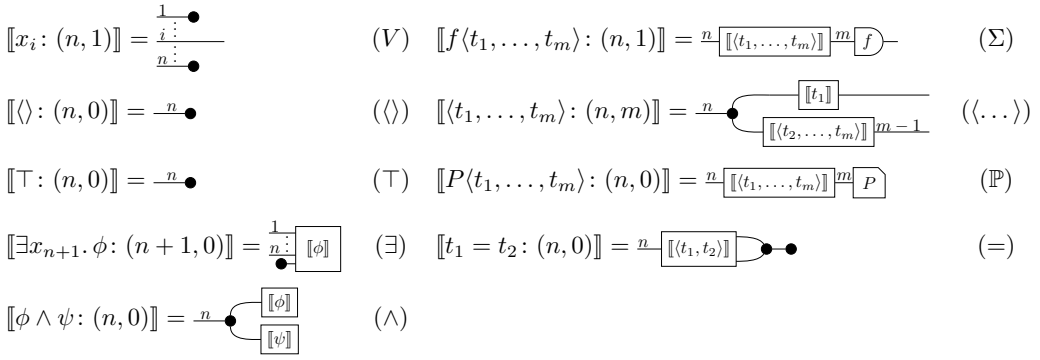
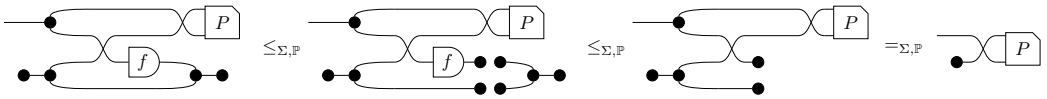


Figure 3 Translation $[\![-]\!]$ from sorted formulas (Figure 1) to string diagrams (Figure 2).

▶ **Example 19.** In Example 8, we outlined the Lindenbaum-Tarski doctrine for the regular fragment of first order logic. We now introduce a cartesian bicategory, denoted by $\mathsf{CB}_{\Sigma,\mathbb{P}}$, that provides a string diagrammatic calculus for this fragment. Like Lawvere theories, $\mathsf{CB}_{\Sigma,\mathbb{P}}$ has $(\mathbb{N},+,0)$ as monoid of objects. Arrows are (equivalence classes of) string diagrams [33] generated according to the rules in Figure 2 (top). For all $n,m \in \mathbb{N}$, identities $\overset{n}{\text{---}}\colon n \to n$, symmetries $\sigma_{n,m}\colon n+m \to m+n$, duplicators $\Delta_n\colon n \to n+n$, dischargers $\varepsilon_n\colon n \to 0$ and their adjoints can be constructed from the basic diagrams in the first row. Observe that function symbols f with arity n are depicted $\overset{n}{\text{---}}\boxed{f}\text{---}$ with coarity 1, while predicate symbols $P \in \mathbb{P}$ with arity n as $\overset{n}{\text{---}}\boxed{P}$ with coarity 0.

Figure 2 (bottom) illustrates the axioms for the calculus: in the last row, function symbols $\overset{n}{\text{---}}\boxed{f}\text{---} : n \to 1$ are forced to be comonoid homomorphism, while predicate symbols $\overset{n}{\text{---}}\boxed{P} : n \to 0$ are just lax; the first three rules impose properties 1, 2 and 3 of Definition 12 on the generating (co)monoid (the laws for arbitrary n follow from these). Let $\leq_{\Sigma,\mathbb{P}}$ be the precongruence with respect to ; and $\otimes$ generated by the axioms and $=_{\Sigma,\mathbb{P}}$ the corresponding equivalence, i.e., $\leq_{\Sigma,\mathbb{P}} \cap \geq_{\Sigma,\mathbb{P}}$. Now $\mathsf{CB}_{\Sigma,\mathbb{P}}[n,m]$ is exactly the set of $=_{\Sigma,\mathbb{P}}$-equivalence classes of diagrams $d\colon n \to m$ ordered by $\leq_{\Sigma,\mathbb{P}}$. Simple inductions suffice to check that $\mathsf{CB}_{\Sigma,\mathbb{P}}$ is indeed a cartesian bicategory and that, moreover, $\mathrm{Map}(\mathsf{CB}_{\Sigma,\mathbb{P}})$ is isomorphic to L_Σ.

In Figure 3 we introduce a function $[\![-]\!]$ that translates sorted formulas $\phi\colon (n,0)$ into string diagrams of type $n \to 0$. From a general result in [32], it follows that ϕ is a logical consequence of ψ if and only if $[\![\psi]\!] \leq_{\Sigma,\mathbb{P}} [\![\phi]\!]$. For an example take $\psi \equiv \exists x_2.\, P(x_2,x_1) \wedge f(x_1) = x_2\colon (1,0)$ and $\phi \equiv \exists x_2.\, P(x_2,x_1)\colon (1,0)$. Then $[\![\psi]\!]$ is the leftmost string diagram below, while $[\![\phi]\!]$ is the rightmost one. The following derivation proves that $\exists x_2.\, P(x_2,x_1)$ is a logical consequence of $\exists x_2.\, P(x_2,x_1) \wedge f(x_1) = x_2$.



5 From Cartesian Bicategories to Doctrines

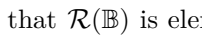
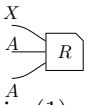
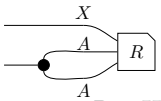
In this section we illustrate how a cartesian bicategory $\mathbb{B}$ gives rise to an elementary existential doctrine. The starting observation is that, using the conclusion of Lemma 13, the functor $\mathrm{Hom}_{\mathbb{B}}(-,I)\colon \mathbb{B}^{\mathrm{op}} \to \mathsf{Set}$ sends objects X to Inf-semilattices. However, for an

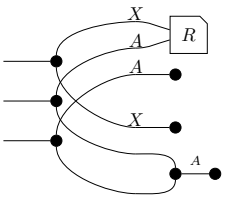
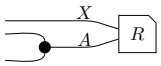
arbitrary morphism $R\colon X \to Y$, $\mathrm{Hom}_{\mathbb{B}}(R, I)\colon \mathrm{Hom}_{\mathbb{B}}(Y, I) \to \mathrm{Hom}_{\mathbb{B}}(X, I)$ may *not* be an inf-preserving function. A sufficient condition is to require R to be a map; indeed, in that case it is immediate to see that infima, as defined in Lemma 13, are preserved. Thus, by restricting the domain of the Hom-functor to $\mathrm{Map}(\mathbb{B})^{\mathrm{op}}$, one obtains a contravariant functor from the cartesian category $\mathrm{Map}(\mathbb{B})$ (Lemma 16) to InfSL:

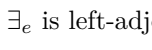
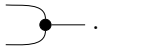
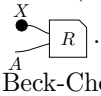
$$\mathcal{R}(\mathbb{B}) = \mathrm{Hom}_{\mathbb{B}}(-, I)\colon \mathrm{Map}(\mathbb{B})^{\mathrm{op}} \to \mathsf{InfSL}\,. \tag{6}$$

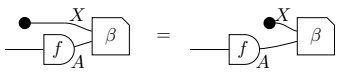
▶ **Theorem 20.** *The functor $\mathcal{R}(\mathbb{B})$ in (6) is an elementary existential doctrine.*

Proof. We need to show that $\mathcal{R}(\mathbb{B})$ is elementary and existential.

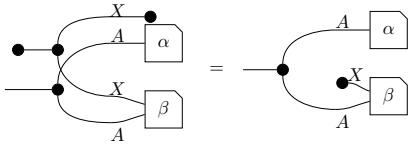
1. First we prove that $\mathcal{R}(\mathbb{B})$ is elementary. We fix $\delta_A =$  $\in \mathrm{Hom}(A \otimes A, I)$. The string diagram for $e = id_X \otimes \Delta_A\colon X \otimes A \to X \otimes A \otimes A$ is  . The function $\mathcal{R}(\mathbb{B})_e$ maps  to  . The function $\exists_e\colon \mathcal{R}(\mathbb{B})(A \otimes X) \to \mathcal{R}(\mathbb{B})(A \otimes X \otimes X)$ defined in (1) maps every $R \in \mathrm{Hom}(X \otimes A, I)$ to

$$\exists_e(R) = \mathcal{R}(\mathbb{B})_{\langle \pi_1, \pi_2 \rangle}(R) \wedge \mathcal{R}(\mathbb{B})_{\langle \pi_2, \pi_3 \rangle}(\delta_A) =$$  $=$ 

   Now $\exists_e$ is left-adjoint to $\mathcal{R}(\mathbb{B})_e$ because  is left-adjoint to  .
2. To show that $\mathcal{R}(\mathbb{B})$ is existential, let $\pi\colon X \otimes A \to A$ be a projection. Then $\exists_\pi$ maps  to  . This is left-adjoint to $\mathcal{R}(\mathbb{B})_\pi$ since  is left-adjoint to  .
   - For the Beck-Chevalley condition, consider the diagram in Remark 7. We need to show that $\exists_{\pi_2}(\mathcal{R}(\mathbb{B})_{\mathrm{id}_X \otimes f}(\beta)) = \mathcal{R}(\mathbb{B})_f(\exists_\pi(\beta))$. Translated to diagrams,

     which holds trivially.
   - For Frobenius reciprocity, take a projection $\pi\colon X \otimes A \to A$, $\alpha \in \mathcal{R}(\mathbb{B})(A)$ and $\beta \in \mathcal{R}(\mathbb{B})(X \otimes A)$. We need that $\exists_\pi(\mathcal{R}(\mathbb{B})_\pi(\alpha) \wedge \beta) = \alpha \wedge \exists_\pi(\beta)$, which translates to

     This holds by naturality of the symmetry and since  is the counit of  . ◀

By applying $\mathcal{R}$ to the cartesian bicategory Rel, one obtains $\mathcal{P}\colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{InfSL}$, in the sense that $\mathcal{P} \cong \mathcal{R}(\mathsf{Rel})$ in EED. The isomorphism is the pair (Γ, b), where $\Gamma\colon \mathsf{Set} \to \mathrm{Map}(\mathsf{Rel})$ is the strict cartesian functor computing the graph of a function and $b\colon \mathcal{P} \to \mathcal{R}(\mathsf{Rel}) \circ \Gamma^{\mathrm{op}}$ is the natural transformation defined for all sets A and $S \in \mathcal{P}(A)$ as $b_A(S) = \{(s, \bullet) \mid s \in S\}$.

▶ **Example 21.** Consider $\mathsf{CB}_{\Sigma,P}$ of Example 19: then $\mathcal{R}(\mathsf{CB}_{\Sigma,P})$ is isomorphic to the doctrine $LT\colon \mathsf{L}_\Sigma^{\mathrm{op}} \to \mathsf{InfSL}$ of Example 8. Here is why: the first two rows in Figure 3 define a cartesian isomorphism $F_{[\![\cdot]\!]}\colon \mathsf{L}_\Sigma \to \mathrm{Map}(\mathsf{CB}_{\Sigma,\mathbb{P}})$. For all $n \in \mathbb{N}$, we define $b_n\colon LT(n) \to$

$\mathrm{Hom}(F_{[\![\cdot]\!]}(n), 0) = \mathrm{Hom}(n, 0)$ as the function mapping $\phi\colon (n, 0)$ to $[\![\phi]\!] : n \to 0$. This give rise to a natural isomorphism $b\colon LT \to \mathcal{R}(\mathsf{CB}_{\Sigma,\mathbb{P}}) \circ F^{\mathrm{op}}_{[\![\cdot]\!]}$. To show that b preserves equalities and existential quantifiers (Definition 9) it suffices to note that $[\![x_1 = x_2]\!] =$  and $[\![\exists x_{n+1}.\, \phi]\!] =$  . The pair $(F_{[\![\cdot]\!]}, b)$ witnesses the isomorphism of LT and $\mathcal{R}(\mathsf{CB}_{\Sigma,\mathbb{P}})$.

▶ **Proposition 22.** *Assigning doctrines to cartesian bicategories as in* (6) *extends to a functor* $\mathcal{R}\colon \mathsf{CBC} \to \mathsf{EED}$. *For a morphism of cartesian bicategories* $F\colon \mathbb{A} \to \mathbb{B}$ *in* CBC, $\mathcal{R}(F) = (F{\restriction_{\mathrm{Map}(\mathbb{A})}}, b^F)$ *where* $b^F\colon \mathcal{R}(\mathbb{A}) \to \mathcal{R}(\mathbb{B}) \circ (F{\restriction_{\mathrm{Map}(\mathbb{A})}})^{\mathrm{op}}$ *is defined as*

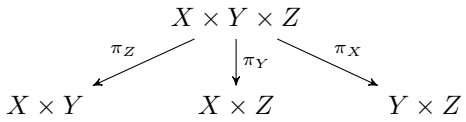
$$b^F_A(U) = F(U) \in \mathrm{Hom}_{\mathbb{B}}(F(A), I) \quad \textit{for all } A \in \mathbb{A} \textit{ and } U \in \mathrm{Hom}_{\mathbb{A}}(A, I). \tag{7}$$

Proof. Let $F\colon \mathbb{A} \to \mathbb{B}$ be a morphism in CBC. By Proposition 18, we have that $F{\restriction_{\mathrm{Map}(\mathbb{A})}}$ is a strict cartesian functor. Regarding b^F, its naturality is ensured by (in fact, equivalent to) the functoriality of F, while the preservation of equalities and existential quantifiers of $\mathcal{R}(\mathbb{A})$ follows from the fact that F preserves the structure of cartesian bicategory of $\mathbb{A}$, which is used to define the structure of elementary existential doctrine of $\mathcal{R}(\mathbb{A})$. Therefore $\mathcal{R}(F)$ is indeed a morphism in EED. Preservation of compositions and identities is straightforward. ◀

6 From Doctrines to Cartesian Bicategories

Given an elementary existential doctrine $P\colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$, we can form a category $\mathscr{A}_P$ whose objects are the same as $\mathbb{C}$ and whose morphisms are given by the elements of $P(X \times Y)$, intuitively seen as *relations*, as observed in [27]. Inspired by the calculus of ordinary relations on sets, using the structure of P we can define a notion of composition and tensor product of these inner relations, and endow each object with a comonoid structure that makes $\mathscr{A}_P$ a cartesian bicategory. Here we recall the essential definitions, while the proof of the fact that $\mathscr{A}_P$ actually satisfies Definition 12 is rather laborious and therefore omitted here: the interested reader can find it in all details in [3].

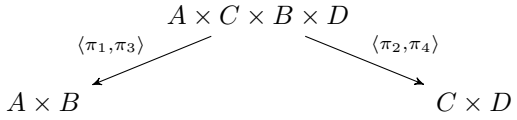
Since $\mathrm{Hom}_{\mathscr{A}_P}(X, Y) = P(X \times Y)$, we get that $\mathscr{A}_P$ is poset-enriched. Given that P is elementary, the obvious candidate for identity on X is $\delta_X \in \mathrm{Hom}_{\mathscr{A}_P}(X, X)$. Composition works as follows: let $f \in \mathrm{Hom}_{\mathscr{A}_P}(X, Y) = P(X \times Y)$ and $g \in \mathrm{Hom}_{\mathscr{A}_P}(Y, Z) = P(Y \times Z)$.



Consider the projections above. Then the composite $f\,;g$ is defined as

$$f\,;g = \exists_{\pi_Y}(P_{\pi_Z}(f) \wedge P_{\pi_X}(g)).$$

The monoidal structure of $\mathscr{A}_P$ is very straightforward: on objects, the monoidal product $\otimes$ is given by the cartesian product in $\mathbb{C}$. On morphisms, for $f \in \mathrm{Hom}_{\mathscr{A}_P}(A, B) = P(A \times B)$ and $g \in \mathrm{Hom}_{\mathscr{A}_P}(C, D) = P(C \times D)$, consider the projections



Then let

$$f \otimes g = P_{\langle \pi_1, \pi_3 \rangle}(f) \wedge P_{\langle \pi_2, \pi_4 \rangle}(g).$$

This makes $\mathscr{A}_P$ a monoidal, poset-enriched category. The rest of the structure of cartesian bicategory is inherited from $\mathbb{C}$ by means of a crucial tool: the *graph functor* of P, hereafter denoted by $\Gamma_P \colon \mathbb{C} \to \mathscr{A}_P$. It is the identity on objects and sends arrows $f \colon X \to Y$ in $\mathbb{C}$ to

$$\Gamma_P(f) = P_{f \times \mathrm{id}_Y}(\delta_Y) \in P(X \times Y) = \mathrm{Hom}_{\mathscr{A}_P}(X, Y).$$

▶ **Proposition 23.** *Let $P \colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ be an elementary, existential doctrine. Then:*

- *Γ_P is strict monoidal,*
- *$\Gamma_P(f)$ has a right adjoint, namely $P_{\mathrm{id}_Y \times f}(\delta_Y)$, for every $f \colon X \to Y$ in $\mathbb{C}$. Therefore, by Lemma 15, it corestricts to a strict cartesian functor $\Gamma_P \colon \mathbb{C} \to \mathrm{Map}(\mathscr{A}_P)$.*

Consider for instance the powerset doctrine $\mathcal{P} \colon \mathsf{Set}^{\mathrm{op}} \to \mathsf{InfSL}$: the elements of $\mathcal{P}(X \times Y)$ are precisely the relations from X to Y, while composition and monoidal product in $\mathscr{A}_{\mathcal{P}}$ coincide with the usual composition and product of relations, see § 4. In other words, $\mathscr{A}_{\mathcal{P}} = \mathsf{Rel}$. The functor $\Gamma_{\mathcal{P}}$ calculates graphs of functions, which are exactly the maps in Rel.

▶ **Example 24.** Recall the doctrine $LT \colon \mathsf{L}_\Sigma \to \mathsf{InfSL}$ and the cartesian bicategory $\mathsf{CB}_{\Sigma,\mathbb{P}}$ from Examples 8 and 19. The functor $\Gamma_{LT} \colon \mathsf{L}_\Sigma \to \mathrm{Map}(\mathscr{A}_{LT})$ is inductively defined as the unique cartesian functor mapping each $f \in \Sigma$ with arity $ar(f) = n$ to the formula $f(x_1, \dots, x_n) = x_{n+1} \colon (n+1, 0)$.

Now, since $\mathbb{C}$ is a cartesian category, every object is canonically equipped with a natural and coherent comonoid structure. We can use the strict monoidal functor $\Gamma_P \colon \mathbb{C} \to \mathscr{A}_P$ to transport this comonoid to $\mathscr{A}_P$, and by Proposition 23 we have that copying and discarding in $\mathscr{A}_P$ both have right adjoints. Finally, one can prove that every morphism in $\mathscr{A}_P$ is a lax-comonoid homomorphism and that the Frobenius law holds.

▶ **Theorem 25.** *Let $P \colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ be an elementary existential doctrine. Then $\mathscr{A}_P$ is a cartesian bicategory. Moreover, the assignment $P \mapsto \mathscr{A}_P$ extends to a functor $\mathcal{L} \colon \mathsf{EED} \to \mathsf{CBC}$ as follows: for P as above, $R \colon \mathbb{D}^{\mathrm{op}} \to \mathsf{InfSL}$ and $(F, b) \colon P \to R$ in EED,*

$$\begin{array}{ccc} \mathscr{A}_P & \xrightarrow{\mathcal{L}(F,b)} & \mathscr{A}_R \\ X & \longmapsto & FX \\ {\scriptstyle P(X\times Y)\ni r}\big\downarrow & \longmapsto & \big\downarrow{\scriptstyle b_{X\times Y}(r)\in R(FX\times FY)} \\ Y & \longmapsto & FY \end{array}$$

7 An Adjunction

We saw in Sections 5 and 6 that using $\mathcal{L}$ and $\mathcal{R}$ one can pass, in a functorial way, between the worlds of cartesian bicategories and elementary existential doctrines. Here we show that, in fact, they define an adjunction $\mathcal{L} \dashv \mathcal{R}$. For this we need natural transformations

$$\eta \colon \mathrm{id}_{\mathsf{EED}} \to \mathcal{R}\mathcal{L}, \qquad \varepsilon \colon \mathcal{L}\mathcal{R} \to \mathrm{id}_{\mathsf{CBC}}$$

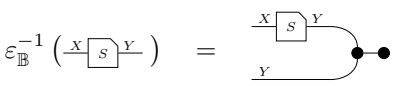
that make the following triangles commute for every $P \colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ in EED and $\mathbb{B}$ in CBC.

$$\begin{array}{ccc} \mathcal{L}(P) & \xrightarrow{\mathcal{L}(\eta_P)} & \mathcal{L}\mathcal{R}\mathcal{L}(P) \\ & {\scriptstyle \mathrm{id}_{\mathcal{L}(P)}}\searrow & \big\downarrow{\scriptstyle \varepsilon_{\mathcal{L}(P)}} \\ & & \mathcal{L}(P) \end{array} \qquad \begin{array}{ccc} \mathcal{R}(\mathbb{B}) & \xrightarrow{\eta_{\mathcal{R}(\mathbb{B})}} & \mathcal{R}\mathcal{L}\mathcal{R}(\mathbb{B}) \\ & {\scriptstyle \mathrm{id}_{\mathcal{R}(\mathbb{B})}}\searrow & \big\downarrow{\scriptstyle \mathcal{R}(\varepsilon_{\mathbb{B}})} \\ & & \mathcal{R}(\mathbb{B}) \end{array} \tag{8}$$

Let us start with ε. Recall that $\mathcal{R}(\mathbb{B}) = \mathrm{Hom}_{\mathbb{B}}(-, I)\colon \mathrm{Map}(\mathbb{B})^{\mathrm{op}} \to \mathsf{InfSL}$ and that, for $f\colon X \to Y$ in $\mathrm{Map}(\mathbb{B})$ and $U \in \mathrm{Hom}_{\mathbb{B}}(Y, I)$, $\mathcal{R}(\mathbb{B})_f(U)$ is equal to $U \circ f\colon X \to I$. By definition then, $\mathcal{LR}(\mathbb{B}) = \mathscr{A}_{\mathcal{R}(\mathbb{B})}$ has the same objects of $\mathbb{B}$ while a morphism $X \to Y$ in $\mathcal{LR}(\mathbb{B})$ is an element of $\mathrm{Hom}_{\mathbb{B}}(X \otimes Y, I)$. Hence, $\varepsilon_{\mathbb{B}}$ has to take a $\begin{smallmatrix} X \\ Y \end{smallmatrix}\boxed{R}$ in $\mathbb{B}$ and produce a morphism $\varepsilon_{\mathbb{B}}(R)\colon X \to Y$ in $\mathbb{B}$. We can do so by "bending" the Y string, defining:

$\varepsilon_{\mathbb{B}}$  (9)

In fact, by the snake equation in (5), $\varepsilon_{\mathbb{B}}$ has an inverse: define, for $S\colon X \to Y$ in $\mathbb{B}$,

$\varepsilon_{\mathbb{B}}^{-1}$ 

and it is immediate to see that $\varepsilon_{\mathbb{B}}^{-1}\varepsilon_{\mathbb{B}}(R) = R$ and $\varepsilon_{\mathbb{B}}\varepsilon_{\mathbb{B}}^{-1}(S) = S$.

▶ **Example 26.** Recall from Example 21 that $\mathcal{R}(\mathsf{CB}_{\Sigma,P}) \cong LT$. Applying $\mathcal{L}$ to both, one gets $\mathcal{LR}(\mathsf{CB}_{\Sigma,P}) \cong \mathcal{L}(LT) = \mathscr{A}_{LT}$, hence using ε we have that $\mathsf{CB}_{\Sigma,P} \cong \mathscr{A}_{LT}$. In particular, given a formula $\phi\colon (n+m, 0)$, one obtains a morphism $n \to m$ in $\mathsf{CB}_{\Sigma,P}$ by first translating ϕ to the string diagram $[\![\phi]\!] : n + m \to 0$ (which is a morphism in $\mathcal{LR}(\mathsf{CB}_{\Sigma,P})$ of type $n \to m$), and then bending the last m inputs using ε as illustrated in (9).

▶ Remark 27. $\varepsilon_{\mathbb{B}}^{-1}$ coincides with $\Gamma_{\mathcal{R}(\mathbb{B})}$ on $\mathrm{Map}(\mathbb{B})$, since $\varepsilon_{\mathbb{B}}^{-1}(f) = \delta_Y^{\mathcal{R}(\mathbb{B})} \circ (f \otimes \mathrm{id}_Y) = \Gamma_{\mathcal{R}(\mathbb{B})}(f)$ when $f\colon X \to Y$ is a map. In other words, $\varepsilon_{\mathbb{B}}^{-1}$ is an extension of Γ_P to the whole of $\mathbb{B}$.

Regarding η, we have $\mathcal{L}(P) = \mathscr{A}_P$, whose objects are the objects of $\mathbb{C}$, and hom-sets are $\mathscr{A}_P(X, Y) = P(X \times Y)$. This means that

$$\mathcal{RL}(P) = \mathrm{Hom}_{\mathscr{A}_P}(-, I) = P(- \times I)\colon \mathrm{Map}(\mathscr{A}_P)^{\mathrm{op}} \to \mathsf{InfSL}\,.$$

To give a morphism $\eta_P\colon P \to \mathcal{RL}(P)$ in EED means therefore to give a functor $F\colon \mathbb{C} \to \mathrm{Map}(\mathscr{A}_P)$ and a natural transformation $b\colon P \to P(F(-) \times I)$ satisfying certain conditions. We have a natural candidate for F: Proposition 23 tells us that Γ_P is a functor whose image, in fact, is included in $\mathrm{Map}(\mathscr{A}_P)$ and, moreover, it is cartesian. Being the identity on objects, the natural transformation part of the definition of η_P must have components $b_X\colon P(X) \to P(X \times I)$: it is clear then that the natural transformation

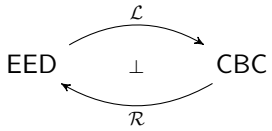
$$P\rho = \big(P_{\rho_X}\colon P(X) \to P(X \times I)\big)_{X \in \mathbb{C}},$$

obtained by whiskering P with the right unitor $\rho_X\colon X \times I \to X$ of $\mathbb{C}$, is a sensible choice for b. In short, we define

$$\eta = \big((\Gamma_P, P\rho)\colon P \to \mathcal{RL}(P)\big)_{P \in \mathsf{EED}}. \tag{10}$$

The interested reader can find a proof of the fact that η and ε are well-defined natural transformations that satisfy the triangular equalities (8) in [3].

▶ **Theorem 28.** *The functors $\mathcal{L}$ and $\mathcal{R}$ form an adjunction*



(11)

whose unit is η (10) and whose counit, which is a natural isomorphism, is ε (9).

Since the counit ε of the adjunction $\mathcal{L} \dashv \mathcal{R}$ is actually a natural isomorphism, the functor $\mathcal{R}\colon \mathsf{CBC} \to \mathsf{EED}$ is full and faithful. It turns out, however, that the adjunction $\mathcal{L} \dashv \mathcal{R}$ is not an equivalence, because $\mathcal{L}$ is not faithful. The following example shows why.

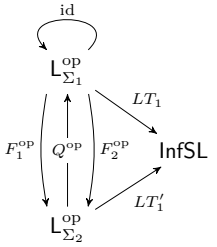
▶ **Example 29.** Let Σ_1 and Σ_2 be the signatures in Example 4 and $\mathbb{P}$ be some signature of predicate symbols. Let $LT_1\colon \mathsf{L}_{\Sigma_1}^{\mathrm{op}} \to \mathsf{InfSL}$ be the indexed Lindenbaum-Tarski algebras for Σ_1 defined as in Example 8. Recall from Example 4 the strict cartesian functor $Q\colon \mathsf{L}_{\Sigma_2} \to \mathsf{L}_{\Sigma_1}$, mapping $g_1, g_2 \in \Sigma_2$ to $f \in \Sigma_1$, and observe that

$$LT_1' = LT_1 \circ Q^{\mathrm{op}}\colon \mathsf{L}_{\Sigma_2}^{\mathrm{op}} \to \mathsf{InfSL}$$

is an elementary, existential doctrine by Remark 11. Its behaviour is somewhat peculiar: it maps $n \in \mathbb{N}$ to the set of formulas $\phi\colon (n, 0)$ built from Σ_1 and $\mathbb{P}$ and to any tuple of terms $\langle t_1, \dots, t_m\rangle\colon n \to m$ in Σ_2 assigns the function mapping a formula $\phi\colon (m, 0)$ to $\phi[{}^{Q(t_1),\dots,Q(t_m)}/_{x_1,\dots,x_m}]$. Observe that each $Q(t_i)$ is again a term in Σ_1: the symbols g_1 and g_2 are both translated to f. Somehow LT_1' behaves like LT_1 but they are different doctrines since their index categories, L_{Σ_2} and L_{Σ_1}, are not isomorphic. However, when transforming them into cartesian bicategories via $\mathcal{L}$, one obtains that $\mathcal{L}(LT_1') = \mathcal{L}(LT_1)$: objects are natural numbers and morphisms $n \to m$ are the elements of the set

$$LT_1'(n+m) = LT_1(Q(n+m)) = LT_1(Q(n) + Q(m)) = LT_1(n+m).$$

To formally show that $\mathcal{L}$ is not faithful we now define two morphisms in EED, both from LT_1 to LT_1', and show that their image along $\mathcal{L}$ is the same. Consider the strict cartesian functors $F_1, F_2\colon \mathsf{L}_{\Sigma_1} \to \mathsf{L}_{\Sigma_2}$ from Example 4 and observe that $QF_i = \mathrm{id}_{\mathsf{L}_{\Sigma_1}}$ for $i = 1, 2$. We are in the following situation:



which means that $LT_1' \circ F_i^{\mathrm{op}} = LT_1 \circ Q^{\mathrm{op}} \circ F_i^{\mathrm{op}} = LT_1$,

thus $(F_1, \mathrm{id}_{LT_1})$ and $(F_2, \mathrm{id}_{LT_1})$ are distinct morphisms in EED from LT_1 to LT_1'. Since $\mathcal{L}(LT_1) = \mathcal{L}(LT_1')$, it follows from the definition of $\mathcal{L}$ that $\mathcal{L}(F_1, \mathrm{id}) = \mathrm{id}_{\mathscr{A}_{LT}} = \mathcal{L}(F_2, \mathrm{id})$.

8 An Equivalence

In the previous section we identified a doctrine with peculiar behaviour, and used it to show that (11) is not an equivalence. Here we characterise the additional constraints on doctrines that are needed for an equivalence.

To make the adjunction (11) an equivalence, we need its unit $\eta\colon \mathrm{id}_{\mathsf{EED}} \to \mathcal{R}\mathcal{L}$ to be a natural isomorphism. This would mean that

$$\eta_P = (\Gamma_P, P\rho)\colon P \to \mathcal{R}\mathcal{L}(P)$$

ought to be invertible in EED for any elementary existential doctrine $P\colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$. Since

$$\mathcal{R}\mathcal{L}(P) = \mathrm{Hom}_{\mathscr{A}_P}(-, I)\colon \mathrm{Map}(\mathscr{A}_P)^{\mathrm{op}} \to \mathsf{InfSL},$$

by definition η_P has an inverse in EED if and only if the functor $\Gamma_P\colon \mathbb{C} \to \mathrm{Map}(\mathscr{A}_P)$ is an isomorphism of cartesian categories, in which case $(\Gamma_P^{-1}, P\rho^{-1})$ would be the inverse of η_P.

But Γ_P is the identity on objects: to be an isomorphism, full and faithful suffices. That is, the maps of $\mathscr{A}_P$ must bijectively correspond with the arrows of the base category $\mathbb{C}$ of P. This is not necessarily the case, as arrows of $\mathbb{C}$ are not involved in the construction of $\mathscr{A}_P$.

The technical tools that allow us to bridge the gap between morphisms in $\mathbb{C}$ and maps in $\mathscr{A}_P$ are provided by [25]: the next two definitions are equivalent to faithfulness and, respectively, fullness of Γ_P.

▶ **Definition 30.** *Let $P\colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ be an elementary existential doctrine and $\alpha \in P(A)$. A* comprehension *of α is an arrow $\{\!|\alpha|\!\}\colon X \to A$ in $\mathbb{C}$ such that $\top_{PX} \leq P_{\{\!|\alpha|\!\}}(\alpha)$ and such that for every $h\colon Z \to A$ for which $\top_{PZ} \leq P_h(\alpha)$, there is a unique arrow $h'\colon Z \to X$ such that $h = \{\!|\alpha|\!\} \circ h'$. P* has comprehensive diagonals *if every diagonal arrow $\Delta_A\colon A \to A \times A$ is the comprehension of δ_A^P.*

▶ **Example 31.** The doctrine $LT_1'\colon \mathsf{L}_{\Sigma_2}^{\mathrm{op}} \to \mathsf{InfSL}$ from Example 29 does not have comprehensive diagonal. Indeed Δ_1 is not the comprehension of δ_1: take as h the arrow $\langle g_1, g_2\rangle\colon 1 \to 2$ and observe that $LT'_{1\langle g_1,g_2\rangle}(\delta_1) = \big(f(x_1) = f(x_1)\big)\colon (1,0)$ and $\top\colon (1,0) \leq \big(f(x_1) = f(x_1)\big)\colon (1,0)$. Yet there exists no $h'\colon 1 \to 1$ in L_{Σ_2} such that $\langle g_1, g_2\rangle = \Delta_1 \circ h'$.

▶ **Definition 32.** *Let $P\colon \mathbb{C}^{\mathrm{op}} \to \mathsf{InfSL}$ be an elementary, existential doctrine. We say that P* satisfies the Rule of Unique Choice *(RUC) if for every $R \in P(X \times Y)$ which is a map in $\mathscr{A}_P$ there exists an arrow $f\colon X \to Y$ such that $\top_{PX} \leq P_{\langle \mathrm{id}_X, f\rangle}(R)$.*

▶ **Example 33.** All the doctrines considered so far satisfy RUC. For an example of a doctrine that does not satisfy it consider the composition of $F_1^{\mathrm{op}}\colon \mathsf{L}_{\Sigma_1}^{\mathrm{op}} \to \mathsf{L}_{\Sigma_2}^{\mathrm{op}}$ (Example 4) and $LT_2\colon \mathsf{L}_{\Sigma_2}^{\mathrm{op}} \to \mathsf{InfSL}$ (Example 8) that, by Remark 11, is an elementary existential doctrine. This doctrine maps n to the set of formulas $\phi\colon (n,0)$ where terms are built from g_1 and g_2, but the index category L_{Σ_1} contains terms built from f that is translated to g_1 by F_1. Now, the formula $\phi \equiv \big(g_2(x_1) = x_2\big)\colon (2,0)$ belongs to $LT_2 \circ F_1^{\mathrm{op}}(1+1)$ and gives rise to a map in $\mathscr{A}_{LT_2 \circ F_1^{\mathrm{op}}}$, but there is no arrow $t\colon 1 \to 1$ in L_{Σ_1} such that $\top \leq LT_2{}_{\langle \mathrm{id}, F_1(t)\rangle}(\phi)$.

We denote by $\overline{\mathsf{EED}}$ the full sub-category of EED consisting only of those elementary existential doctrines with comprehensive diagonals and satisfying the Rule of Unique Choice. Conveniently, it turns out that the image of $\mathcal{R}$ is already contained in $\overline{\mathsf{EED}}$.

▶ **Proposition 34.** *Let $\mathbb{B}$ be a cartesian bicategory. Then $\mathcal{R}(\mathbb{B})$ is in $\overline{\mathsf{EED}}$.*

Proof. It is enough to show that $\Gamma_{\mathcal{R}(\mathbb{B})}\colon \mathrm{Map}(\mathbb{B}) \to \mathrm{Map}(\mathscr{A}_{\mathcal{R}(\mathbb{B})}) = \mathrm{Map}(\mathcal{L}\mathcal{R}(\mathbb{B}))$ is full and faithful. As noticed in Remark 27, $\Gamma_{\mathcal{R}(\mathbb{B})} = \varepsilon_{\mathbb{B}}^{-1}{\restriction}_{\mathrm{Map}(\mathbb{B})}$. Since $\varepsilon_{\mathbb{B}}^{-1}$ is an isomorphism in CBC, it is faithful, therefore its restriction $\Gamma_{\mathcal{R}(\mathbb{B})}$ is as well. Moreover, $\varepsilon_{\mathbb{B}}^{-1}$ is full: if $R\colon X \to Y$ in $\mathcal{L}\mathcal{R}(\mathbb{B})$ is a map, then there exists $f\colon X \to Y$ in $\mathbb{B}$ such that $\varepsilon_{\mathbb{B}}^{-1}(f) = R$. In fact, $f = \varepsilon_{\mathbb{B}}(R)$ and since $\varepsilon_{\mathbb{B}}$ is a morphism in CBC, by Proposition 18 we have that f is a map in $\mathbb{B}$. Therefore, $\Gamma_{\mathcal{R}(\mathbb{B})}$ is full. ◀

▶ **Theorem 35.** *The categories $\overline{\mathsf{EED}}$ and CBC are equivalent via adjunction (11) where $\mathcal{L}$ and $\mathcal{R}$ are respectively restricted and corestricted to $\overline{\mathsf{EED}}$.*

9 Conclusion

We gave an exhaustive analysis of the relationship between two different categorifications of regular logic: the *universal* approach of elementary existential doctrines, and the *algebraic* approach of cartesian bicategories. We showed that cartesian bicategories give rise to elementary existential doctrines and, expanding a remark in [27], that also the other direction is possible. We proved that this correspondence is functorial, in the sense that we have a pair of functors $\mathcal{L}\colon \mathsf{EED} \to \mathsf{CBC}$ and $\mathcal{R}\colon \mathsf{CBC} \to \mathsf{EED}$ which are moreover adjoint (Theorem 28).

This adjunction can be strengthened to an equivalence provided that we refine the notion of doctrine, excluding some problematic examples (e.g. Example 29). These cases lay outside the image of $\mathcal{R}$ and thus this restriction does not affect cartesian bicategories (Theorem 35).

We hope that understanding the relationship between CBC and EED may provide some hints on the nature of the additional algebraic structure needed for cartesian bicategories to capture full first order logic, which one can do on the EED, universal side by considering Lawvere's original hyperdoctrines. It is probable that the end result will be closely related to Peirce's existential graphs [31], a 19th century proto-string-diagrammatic logical syntax. This direction has already started to be explored, from diverse perspectives, in [7, 8, 18].

References

1 John Baez and Jason Erbele. Categories in control. *Theory and Application of Categories*, 30(24):836–881, 2015. URL: http://www.tac.mta.ca/tac/volumes/30/24/30-24abs.html.

2 Filippo Bonchi, Joshua Holland, Robin Piedeleu, Pawel Sobocinski, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. *Proc. ACM Program. Lang.*, 3(POPL):25:1–25:28, 2019. doi:10.1145/3290338.

3 Filippo Bonchi, Alessio Santamaria, Jens Seeber, and Paweł Sobociński. On Doctrines and Cartesian Bicategories. *arXiv:2106.08142 [cs, math]*, June 2021. arXiv:2106.08142.

4 Filippo Bonchi, Jens Seeber, and Pawel Sobocinski. Graphical Conjunctive Queries. In Dan Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic (CSL 2018)*, volume 119 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:23, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2018.13.

5 Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Full Abstraction for Signal Flow Graphs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '15, pages 515–526, New York, NY, USA, January 2015. Association for Computing Machinery. doi:10.1145/2676726.2676993.

6 Filippo Bonchi, Pawel Sobocinski, and Fabio Zanasi. Deconstructing Lawvere with distributive laws. *Journal of Logical and Algebraic Methods in Programming*, 95:128–146, 2018. doi:10.1016/j.jlamp.2017.12.002.

7 Geraldine Brady and Todd Trimble. A string diagram calculus for predicate logic and C. S. Peirce's system beta. Unpublished, available online at https://ncatlab.org/nlab/files/BradyTrimbleString.pdf, 2000.

8 Geraldine Brady and Todd H. Trimble. A categorical interpretation of C.S. Peirce's propositional logic Alpha. *Journal of Pure and Applied Algebra*, 149(3):213–239, June 2000. doi:10.1016/S0022-4049(98)00179-0.

9 Aurelio Carboni and Robert F.C. Walters. Cartesian Bicategories I. *Journal of Pure and Applied Algebra*, 49(1):11–32, 1987. doi:10.1016/0022-4049(87)90121-6.

10 Bob Coecke and Ross Duncan. Interacting quantum observables: categorical algebra and diagrammatics. *New Journal of Physics*, 13(4):043016, April 2011. Publisher: IOP Publishing. doi:10.1088/1367-2630/13/4/043016.

11 Ivan Di Liberti, Fosco Loregian, Chad Nester, and Paweł Sobociński. Functorial semantics for partial theories. In *Proceedings of the ACM on Programming Languages*, volume 5, pages 57:1–57:28, January 2021. doi:10.1145/3434338.

12 Brendan Fong, Paweł Sobociński, and Paolo Rapisarda. A categorical approach to open and interconnected dynamical systems. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '16, pages 495–504, New York, NY, USA, July 2016. Association for Computing Machinery. doi:10.1145/2933575.2934556.

13 Brendan Fong and David Spivak. String diagrams for regular logic (extended abstract). In John Baez and Bob Coecke, editors, *Applied Category Theory 2019*, volume 323 of *Electronic Proceedings in Theoretical Computer Science*, pages 196–229. Open Publishing Association, September 2020. doi:10.4204/eptcs.323.14.

14 Brendan Fong and David I Spivak. Graphical regular logic, 2019. arXiv:1812.05765.

15 Thomas Fox. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7):665–667, 1976. doi:10.1080/00927877608822127.

16 Peter Freyd and Andre Scedrov. *Categories, Allegories*, volume 39 of *North-Holland Mathematical Library*. Elsevier B.V, 1990. URL: https://www.sciencedirect.com/bookseries/north-holland-mathematical-library/vol/39/suppl/C.

17 Dan R. Ghica and Achim Jung. Categorical semantics of digital circuits. In *2016 Formal Methods in Computer-Aided Design (FMCAD)*, pages 41–48, October 2016. doi:10.1109/FMCAD.2016.7886659.

18 Nathan Haydon and Paweł Sobociński. Compositional Diagrammatic First-Order Logic. In *Diagrammatic Representation and Inference*, Lecture Notes in Computer Science, 11th International Conference, Diagrams 2020, Tallinn, Estonia,, 2020. Springer International Publishing. doi:10.1007/978-3-030-54249-8_32.

19 Stephen Lack. Composing PROPs. *Theory and Application of Categories*, 13(9):147–163, 2004. URL: http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html.

20 F. William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences*, 50(5):869–872, 1963.

21 F. William Lawvere. Adjointness in Foundations. *Dialectica*, 23(3/4):281–296, 1969. Publisher: Wiley. doi:10.1111/j.1746-8361.1969.tb01194.x.

22 F. William Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications II*, Lecture Notes in Mathematics, pages 134–145, Berlin, Heidelberg, 1969. Springer. doi:10.1007/BFb0080769.

23 F. William Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. In Alex Heller, editor, *Applications of Categorical Algebra*, volume 17, pages 1–14, New York, NY, 1970. American Mathematical Society. doi:10.1090/pspum/017.

24 Saunders MacLane. Categorical algebra. *Bulletin of the American Mathematical Society*, 71(1):40–106, 1965. doi:10.1090/S0002-9904-1965-11234-4.

25 Maria Emilia Maietti, Fabio Pasquali, and Giuseppe Rosolini. Triposes, exact completions, and Hilbert's ε-operator. *Tbilisi Mathematical Journal*, 10(3):141–166, June 2017. Publisher: Tbilisi Centre for Mathematical Sciences. doi:10.1515/tmj-2017-0106.

26 Maria Emilia Maietti and Giuseppe Rosolini. Quotient Completion for the Foundation of Constructive Mathematics. *Logica Universalis*, 7(3):371–402, September 2013. doi:10.1007/s11787-013-0080-2.

27 Maria Emilia Maietti and Giuseppe Rosolini. Unifying Exact Completions. *Applied Categorical Structures*, 23(1):43–52, February 2015. doi:10.1007/s10485-013-9360-5.

28 Koko Muroya, Steven W. T. Cheung, and Dan R. Ghica. The geometry of computation-graph abstraction. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 749–758. ACM, 2018. doi:10.1145/3209108.3209127.

29 Evan Patterson. Knowledge representation in bicategories of relations, 2017. arXiv:1706.00526.

30 Robin Piedeleu and Fabio Zanasi. A String Diagrammatic Axiomatisation of Finite-State Automata. In Stefan Kiefer and Christine Tasson, editors, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science, pages 469–489, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-71995-1_24.

31 Ahti-Veikko Pietarinen. *The Logic of the Future*, volume 1. De Gruyter, 2019.

32 Jens Seeber. *Logical completeness for string diagrams*. PhD thesis, IMT Lucca, 2020.

33 P. Selinger. A Survey of Graphical Languages for Monoidal Categories. In B. Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 289–355. Springer, Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-12821-9_4.

Presenting Convex Sets of Probability Distributions by Convex Semilattices and Unique Bases

Filippo Bonchi
University of Pisa, Italy

Ana Sokolova
University of Salzburg, Austria

Valeria Vignudelli
Univ Lyon, CNRS, ENS Lyon, UCB Lyon 1, LIP, France

Abstract

We prove that every finitely generated convex set of finitely supported probability distributions has a unique base. We apply this result to provide an alternative proof of a recent result: the algebraic theory of convex semilattices presents the monad of convex sets of probability distributions.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Axiomatic semantics; Theory of computation → Categorical semantics

Keywords and phrases Convex sets of distributions monad, Convex semilattices, Unique base

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.11

Category (Co)algebraic pearls

Funding *Filippo Bonchi*: Supported by the Ministero dell'Università e della Ricerca of Italy under Grant No. 201784YSZ5, PRIN2017 – ASPRA (Analysis of Program Analyses).
Valeria Vignudelli: Supported by the French projects ANR-20-CE48-0005 QuaReMe and ANR-16-CE25-0011 REPAS, the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157), the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

1 Introduction

Models of computations exhibiting both nondeterministic and probabilistic behaviour are abundantly used in computed assisted verification [1, 12, 19, 5, 35, 11, 27], Artificial Intelligence [4, 17, 26], and studied from semantics perspective [14, 29, 13]. Indeed, probability is needed to quantitatively model uncertainty and belief, whereas nondeterminism enables modelling of incomplete information, unknown environment, implementation freedom, or concurrency.

Since several decades, computer scientists have found it convenient to exploit algebraic methods to analyse computing systems. From an algebraic perspective, the interplay of nondeterminism and probability has been posing some remarkable challenges [34, 18, 20, 16, 33, 24, 9, 31, 23]. Nevertheless, several fundamental algebraic structures have been identified and studied in depth.

In this paper we focus on one such structure, namely *convex sets of probability distributions*. These sets give rise to a monad that is well known in the literature and has found applications in several works [24, 9, 31, 33, 34, 16, 10, 22]. In recent work [3], we proved that this monad is presented by the algebraic theory of *convex semilattices*. In this paper, we provide an alternative proof based on a simple property: We show that every (finitely generated) convex set of distributions has a *unique base*.

This alternative proof technique is based on a categorical machinery together with a more syntax-based approach, which has already proven useful in extensions of the presentation results to the setting of metric spaces and quantitative equational theories [22, 21].

Synopsis. In Section 2, we show the unique base theorem. Our alternative proof of the presentation of the monad is based on exhibiting a monad map which is an isomorphism. We recall the relevant categorical notions in Section 3, and introduce a general recipe for building a monad map. In Section 4 we illustrate the monad of interest as well as the theory of convex semilattices, and in Section 5 we apply the recipe from Section 3 to build a monad map relating the monad and the theory. In Section 6 we prove that this monad map is an isomorphism, by relying on the unique base theorem to derive a normal-form argument.

2 A unique base theorem for convex sets of probability distributions

Given a set X, a probability distribution is a function $d\colon X \to [0,1]$ such that $\sum_{x\in X} d(x) = 1$. A probability distribution d is finitely supported if $d(x) \neq 0$ for finitely many x. We call $\mathcal{D}(X)$ the set of finitely suported probability distributions over X. A probability distribution $d \in \mathcal{D}(X)$ is a convex combination of the distributions $d_1, \dots d_n \in \mathcal{D}(X)$ if there exist $\alpha_1, \dots, \alpha_n \in [0,1]$ such that $\sum_i \alpha_i = 1$ and for all x, $d(x) = \sum_i \alpha_i d_i(x)$. Hereafter we will just write the latter condition as $d = \sum_i \alpha_i d_i$. The *convex closure* of a subset $S \subseteq \mathcal{D}(X)$, written $\mathrm{conv}(S)$, is the set of all the convex combinations of the distributions in S. A subset $S \subseteq \mathcal{D}(X)$ is *convex* if $S = \mathrm{conv}(S)$. A convex set is *finitely generated* if there exist $d_1, \dots, d_n \in \mathcal{D}(X)$ such that $S = \mathrm{conv}(\{d_1, \dots, d_n\})$. We let $C(X)$ denote the set of non-empty, finitely-generated convex sets of distributions over X. A *base* for $S \in C(X)$ is a set $\{d_1, \dots, d_n\}$ such that $S = \mathrm{conv}(\{d_1, \dots, d_n\})$ and for all $i \in 1 \dots n$, $d_i \notin \mathrm{conv}(\{d_j \mid j \neq i, 1 \leq j \leq n\})$.

▶ **Theorem 1.** *For every $S \in C(X)$, there exists a unique base.*

We show here a direct proof (Proof I) and an alternative proof using functional analysis tools and the strong theorem of Krein-Milman [25] (Proof II).

Proof I. Existence of the base comes from the property that S is finitely generated. In the rest of this section we prove uniqueness; namely if $\{d_1, \dots, d_n\}$ and $\{d'_1, \dots, d'_m\}$ are two bases for some $S \in \mathcal{D}(X)$, then $\{d_1, \dots, d_n\} = \{d'_1, \dots, d'_m\}$.

Let $\{d_1, \dots, d_n\}$ and $\{d'_1, \dots, d'_m\}$ be two bases for $S \in \mathcal{D}(X)$. Then for all $i \in 1 \dots n$ it holds $d_i \in \mathrm{conv}(\{d'_1, \dots, d'_m\})$ and for all $j \in 1 \dots m$ it holds $d'_j \in \mathrm{conv}(\{d_1, \dots, d_n\})$. By unfolding the definition of conv, this means that for all i there exist $\alpha_{i,j}$ such that $\sum_j \alpha_{i,j} = 1$ and for all j there exist $\alpha'_{j,i}$ such that $\sum_i \alpha'_{j,i} = 1$ and such that

$$d_i = \sum_{j\in\{1\dots m\}} \alpha_{i,j} d'_j \quad \text{and} \quad d'_j = \sum_{i\in\{1\dots n\}} \alpha'_{j,i} d_i. \tag{1}$$

Hence, for all i it holds

$$d_i = \sum_{j\in\{1\dots m\}} \alpha_{i,j} \left(\sum_{k\in\{1\dots n\}} \alpha'_{j,k} d_k \right) = \sum_{k\in\{1\dots n\}} \left(\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,k} \right) d_k$$

where the fist equality follows by replacing the d'_j in the left equation in (1) with the one in the right equation in (1). So we have

$$d_i = \left(\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,i}\right) d_i \;+\; \sum_{k\in\{1\dots n\}\setminus\{i\}} \left(\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,k}\right) d_k \tag{2}$$

We now prove by contradiction that

$$\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,i} = 1 \text{ for all } i \in \{1\dots n\} \tag{3}$$

Let $i \in \{1\dots n\}$ and let $\beta_i = \sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,i}$. If $\beta_i \neq 1$, then by (2) we have

$$d_i = \beta_i d_i \;+\; (1-\beta_i) \sum_{k\in\{1\dots n\}\setminus\{i\}} \left(\frac{\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,k}}{1-\beta_i}\right) d_k$$

and from this we derive

$$d_i = \sum_{k\in\{1\dots n\}\setminus\{i\}} \left(\frac{\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,k}}{1-\beta_i}\right) d_k$$

This means that d_i is expressible as a convex combination of $\{d_1\dots,d_n\}\setminus\{d_i\}$, which contradicts the hypothesis that $\{d_1\dots,d_n\}$ is a base. Hence, $\beta_i = 1$, which proves (3).

From (3) and (2) we derive that for all $k \in \{1\dots n\}\setminus\{i\}$, $\sum_{j\in\{1\dots m\}} \alpha_{i,j}\alpha'_{j,k} = 0$. Since all the summands are non-negative, this entails that

$$\alpha_{i,j}\alpha'_{j,k} = 0 \text{ for all } i \in \{1\dots n\},\ k \in \{1\dots n\}\setminus\{i\} \text{ and } j \in \{1\dots m\}. \tag{4}$$

By reasoning in the same way, we obtain the following

$$\alpha'_{j,i}\alpha_{i,l} = 0 \text{ for all } j \in \{1\dots m\},\ l \in \{1\dots m\}\setminus\{j\} \text{ and } i \in \{1\dots n\}. \tag{5}$$

We now prove that for all i there exists one j such that $\alpha_{i,j} = 1$. As $\sum_j \alpha_{i,j} = 1$, there is at least one j such that $\alpha_{i,j} > 0$. By this and (4) one has that for all $k \in \{1\dots n\}\setminus\{i\}$, $\alpha'_{j,k} = 0$. Since $\sum_{k\in\{1\dots n\}} \alpha'_{j,k} = 1$, we have that $\alpha'_{j,i} = 1$. Hence we derive by (5) that $\alpha_{i,l} = 0$ for all $l \in \{1\dots m\}\setminus\{j\}$. Since $\sum_{l\in\{1\dots m\}} \alpha_{i,l} = 1$, we have $\alpha_{i,j} = 1$.

Using this fact, we conclude by the left equation in (1) that for every i there exists one j such that $d_i = d'_j$. Hence, we have $\{d_1,\dots,d_n\} \subseteq \{d'_1,\dots,d'_m\}$. The opposite inclusion follows symmetrically. ◀

Proof II. Let $S \in C(X)$. Note that then S is a subset of $\mathcal{D}(X) \subseteq \mathbb{R}^X$ and hence a subset of a locally convex topological vector space ($\mathbb{R}^X$ with the product topology). Consider the family $\mathcal{B} = \{B \subseteq S \mid S = \mathrm{conv}(B)\}$. It is obvious that B is minimal in $\mathcal{B}$ if and only if no element $d \in B$ satisfies $d \in \mathrm{conv}(B\setminus\{d\})$. We now show that $\mathcal{B}$ contains a smallest element.

First, note that for all $B \in \mathcal{B}$, $\mathrm{Ext}(S) \subseteq B$, with $\mathrm{Ext}(S)$ being the set of extreme points of S. Indeed, let $d \in \mathrm{Ext}(S)$. Then $d \in S$ and can be written as $d = \sum_{d_i\in B} p_i d_i = p_i \cdot d_i + (1-p_i)\cdot e$ for some $p_i \neq 0$ and $e \in S$, and hence by extremality of d we have $d = d_i = e$ yielding $d \in B$.

Next, we show that $S = \mathrm{conv}(\mathrm{Ext}(S))$, which means that $\mathrm{Ext}(S) \in \mathcal{B}$ and hence together with $\mathrm{Ext}(S) \subseteq B$ shows that $\mathrm{Ext}(S)$ is the smallest element of $\mathcal{B}$. This smallest element $\mathrm{Ext}(S)$ is the unique base of S. Pick a finite $B_0 = \{d_1,\dots,d_n\} \in \mathcal{B}$. Then $S = \Phi(\Delta_n)$ for

$$\Delta_n = \{(x_1,\dots,x_n) \in \mathbb{R}^n \mid x_i \in [0,1], \sum_i x_i = 1\}$$

and $\Phi\colon \mathbb{R}^n \to \mathbb{R}^X$ given by $\Phi(x_1,\dots,x_n) = \sum_i x_i d_i$. Note that Δ_n is compact, by Heine-Borel, as it is a closed and bounded subset of $\mathbb{R}^n$, and Φ is continuous, since we are in a topological vector space and hence algebraic operations are continuous. As a consequence, S is compact as a continuous image of a compact set. Now, Krein-Milman applies, yielding that $S = \overline{\text{conv}}(\text{Ext}(S))$ with $\overline{\text{conv}}$ denoting the closed convex hull and hence

$$S = \overline{\text{conv}}(\text{Ext}(S)) = \text{conv}(\text{Ext}(S))$$

since by the same argument as above $\text{conv}(\text{Ext}(S))$ is compact and hence closed. ◀

Instead of the Krein-Milman theorem, one could use in this proof its predecessor from classical convex analysis in $\mathbb{R}^n$, e.g. [32, Theorem 18.5]. The reason is that since we deal with finitely generated convex subsets of finitely supported distributions, such subsets are actually elements of $C(X)$ for a finite set X.

3 Monads and presentations

Theorem 1 states the existence of a unique base for every finitely generated convex set of probability distributions. In the remainder of this paper, we exploit this result to illustrate an alternative proof of Theorem 4 in [3] that provides a presentation of the monad C [24, 9, 31, 33, 34, 16]. In Section 4, we recall the monad as well as its presentation given in [3]. In this section, we recall some basic facts about monads and presentations.

A *monad* on **Sets** is a functor $\mathcal{M}\colon \mathbf{Sets} \to \mathbf{Sets}$ together with two natural transformations: a unit $\eta\colon \text{Id} \Rightarrow \mathcal{M}$ and multiplication $\mu\colon \mathcal{M}^2 \Rightarrow \mathcal{M}$ that satisfy the laws $\mu \circ \eta\mathcal{M} = \mu \circ \mathcal{M}\eta = id$ and $\mu \circ \mathcal{M}\mu = \mu \circ \mu\mathcal{M}$.

A *monad map* from a monad $\mathcal{M}$ to a monad $\hat{\mathcal{M}}$ is a natural transformation $\sigma\colon \mathcal{M} \Rightarrow \hat{\mathcal{M}}$ that makes the following diagrams commute, with η, μ and $\hat{\eta}, \hat{\mu}$ denoting the unit and multiplication of $\mathcal{M}$ and $\hat{\mathcal{M}}$, respectively, and $\sigma\sigma = \sigma \circ \mathcal{M}\sigma = \hat{\mathcal{M}}\sigma \circ \sigma_{\mathcal{M}}$.

$$\begin{array}{ccc} X & \xrightarrow{\eta} & \mathcal{M}X \\ & \searrow^{\hat{\eta}} & \downarrow{\scriptstyle \sigma} \\ & & \hat{\mathcal{M}}X \end{array} \qquad \begin{array}{ccc} \mathcal{M}\mathcal{M}X & \xrightarrow{\sigma\sigma} & \hat{\mathcal{M}}\hat{\mathcal{M}}X \\ {\scriptstyle \mu}\downarrow & & \downarrow{\scriptstyle \hat{\mu}} \\ \mathcal{M}X & \xrightarrow[\sigma]{} & \hat{\mathcal{M}}X \end{array}$$

If $\sigma\colon \mathcal{M}X \to \hat{\mathcal{M}}X$ is an iso, the two monads are isomorphic.

An important example of monad is provided by the *free monad of terms*. Given a signature Σ, namely a set of operation symbols equipped with an arity, the free monad $T_\Sigma\colon \mathbf{Sets} \to \mathbf{Sets}$ of terms over Σ maps a set X to the set of all Σ-terms with variables in X, and $f\colon X \to Y$ to the function that maps a term over X to a term over Y obtained by substitution according to f. The unit maps a variable in X to itself, and the multiplication is term composition.

Given a set of axioms E over Σ-terms, one can define the smallest congruence generated by the axioms, denoted by $=_E$. Hereafter we write $[t]_E$ for the $=_E$-equivalence class of the Σ-term t and $T_{\Sigma,E}(X)$ for the set of E-equivalence classes of Σ-terms with variables in X. The assignment $X \mapsto T_{\Sigma,E}(X)$ gives rise to a functor $T_{\Sigma,E}\colon \mathbf{Sets} \to \mathbf{Sets}$ where the behaviour on functions is defined as for T_Σ. Such functor carries the structure of a monad: the unit $\eta^E\colon \text{Id} \Rightarrow T_{\Sigma,E}$ and the multiplication $\mu^E\colon T_{\Sigma,E}T_{\Sigma,E} \Rightarrow T_{\Sigma,E}$ are defined as $\eta^E(x) = [x]_E$ and $\mu^E[t\{[t_i]_E/x_i\}]_E = [t\{t_i/x_i\}]_E$.

An *algebraic theory* is a pair (Σ, E) of signature Σ and a set of equations E. We say that (Σ, E) provides a *presentation* for a monad $\mathcal{M}$ if $T_{\Sigma,E}$ is isomorphic to $\mathcal{M}$.

We next introduce several monads on **Sets** together with their presentations.

Nondeterminism. The non-empty finite powerset monad $\mathcal{P}_{ne}$ maps a set X to the set of non-empty finite subsets $\mathcal{P}_{ne}X = \{U \mid U \subseteq X,\ U \text{ is finite and non-empty}\}$ and a function $f\colon X \to Y$ to $\mathcal{P}_{ne}f\colon \mathcal{P}_{ne}X \to \mathcal{P}_{ne}Y$, $\mathcal{P}_{ne}f(U) = \{f(u) \mid u \in U\}$. The unit η of $\mathcal{P}_{ne}$ is given by singleton, i.e., $\eta(x) = \{x\}$, and the multiplication μ is given by union, i.e., $\mu(S) = \bigcup_{U \in S} U$ for $S \in \mathcal{P}_{ne}\mathcal{P}_{ne}X$.

Let Σ_N be the signature consisting of a binary operation $\oplus$. Let E_N be the following set of axioms, the axioms of semilattice:

$$(x \oplus y) \oplus z \overset{(A)}{=} x \oplus (y \oplus z) \qquad x \oplus y \overset{(C)}{=} y \oplus x \qquad x \oplus x \overset{(I)}{=} x$$

It is easy to show that the algebraic theory (Σ_N, E_N) provides a presentation for the monad $\mathcal{P}_{ne}$, in the sense that there exists an isomorphism of monads $\iota^N\colon T_{\Sigma_N,E_N} \Rightarrow \mathcal{P}_{ne}$.

Probability. The finitely supported probability distribution monad $\mathcal{D}$ is defined, for a set X and a function $f\colon X \to Y$, as $\mathcal{D}X = \{\varphi\colon X \to [0,1] \mid \sum_{x \in X} \varphi(x) = 1,\ \mathrm{supp}(\varphi) \text{ is finite}\}$ and $\mathcal{D}f(\varphi)(y) = \sum_{x \in f^{-1}(y)} \varphi(x)$. The unit of $\mathcal{D}$ is given by a Dirac distribution $\eta(x) = \delta_x = (x \mapsto 1)$ for $x \in X$ and the multiplication by $\mu(\Phi)(x) = \sum_{\varphi \in \mathrm{supp}(\Phi)} \Phi(\varphi) \cdot \varphi(x)$ for $\Phi \in \mathcal{D}\mathcal{D}X$. We sometimes write $\sum_{i \in I} p_i x_i$ for a distribution φ with $\mathrm{supp}(\varphi) = \{x_i \mid i \in I\}$ and $\varphi(x_i) = p_i$.

Let Σ_P be the signature consisting of a binary operation $+_p$ for all $p \in (0,1)$. Let E_P be the following set of axioms, the axioms of a barycentric algebra, also called convex algebra:[1]

$$(x +_q y) +_p z \overset{(A_p)}{=} x +_{pq} (y +_{\frac{p(1-q)}{1-pq}} z) \qquad x +_p y \overset{(C_p)}{=} y +_{1-p} x \qquad x +_p x \overset{(I_p)}{=} x$$

The algebraic theory (Σ_P, E_P) provides a presentation for the monad $\mathcal{D}$ [30, 28, 7, 8, 15], in the sense that there exists an isomorphism of monads $\iota^P\colon T_{\Sigma_P,E_P} \Rightarrow \mathcal{D}$.

3.1 A well known recipe for constructing monad morphisms

To prove that an algebraic theory (Σ, E) presents a monad $\mathcal{M}$, one has to provide $\iota\colon T_{\Sigma,E} \Rightarrow \mathcal{M}$ that (a) is a monad map and (b) is an isomorphism. While the proof of (b) often requires some specific normal form arguments, the proof of (a) can be significantly simplified by using some standard categorical machinery.

In this section, we illustrate a well known recipe which allows for constructing a monad map $\iota\colon T_{\Sigma,E} \Rightarrow \mathcal{M}$ in a principled way. We begin by recalling Eilenberg-Moore algebras.

To each monad $\mathcal{M}$, one associates the Eilenberg-Moore category $\mathrm{EM}(\mathcal{M})$ of $\mathcal{M}$-algebras. Objects of $\mathrm{EM}(\mathcal{M})$ are pairs $\mathbb{A} = (A, a)$ of a set $A \in \mathbf{Sets}$ and a map $a\colon \mathcal{M}A \to A$, making the first two diagrams below commute.

$$\begin{array}{ccc} A & \xrightarrow{\eta} & \mathcal{M}A \\ & \searrow & \downarrow a \\ & & A \end{array} \qquad \begin{array}{ccc} \mathcal{M}^2A & \xrightarrow{\mathcal{M}a} & \mathcal{M}A \\ {\scriptstyle\mu}\downarrow & & \downarrow{\scriptstyle a} \\ \mathcal{M}A & \xrightarrow[a]{} & A \end{array} \qquad \begin{array}{ccc} \mathcal{M}A & \xrightarrow{\mathcal{M}h} & \mathcal{M}B \\ {\scriptstyle a}\downarrow & & \downarrow{\scriptstyle b} \\ A & \xrightarrow[h]{} & B \end{array}$$

A homomorphism from an algebra $\mathbb{A} = (A, a)$ to an algebra $\mathbb{B} = (B, b)$ is a map $h\colon A \to B$ between the underlying sets making the third diagram above commute.

[1] There is another equivalent presentation for convex algebras with a signature involving arbitrary convex combinations and two axioms, projection and barycenter. In this paper we will mainly use the binary convex operations.

It is well known that, when $\mathcal{M}$ is the monad $T_{\Sigma,E}$ for some algebraic theory (Σ, E), $\mathrm{EM}(\mathcal{M})$ is isomorphic to the category $\mathrm{Alg}(\Sigma, E)$ of (Σ, E)-algebras and their morphisms. A Σ-algebra (X, Σ_X) consist of a set X together with a set Σ_X of operations $\hat{o}_X \colon X^n \to X$, one for each operation symbol $o \in \Sigma$ of arity n. A (Σ, E)-algebra is a Σ-algebra where all the equations in E hold. A homomorphism h from a (Σ, E)-algebra (X, Σ_X) to a (Σ, E)-algebra (Y, Σ_Y) is a function $h \colon X \to Y$ that commutes with the operations, i.e., $h \circ \hat{o}_X = \hat{o}_Y \circ h^n$ for all n-ary $o \in \Sigma$.

For instance, (Σ_N, E_N)-algebras are semilattices, namely a set X equipped with a binary operation $\hat{\oplus}_X$ that is associative, commutative and idempotent. A semilattice homomorphism is a function $h \colon X \to Y$ such that $h(x_1 \hat{\oplus}_X x_2) = h(x_1) \hat{\oplus}_Y h(x_2)$ for all $x_1, x_2 \in X$.

Now we can display an abstract recipe for constructing a monad map $\iota \colon T_{\Sigma,E} \Rightarrow \mathcal{M}$, which consists of three steps:

(A) For each set X, provide $\mathcal{M}X$ with the structure of a (Σ, E)-algebra, namely functions $\hat{o}_X \colon (\mathcal{M}X)^n \to \mathcal{M}X$ for each $o \in \Sigma$, that satisfy the equations in E;

(B) Prove that for each function $f \colon X \to Y$, $\mathcal{M}f$ is a (Σ, E)-algebra homomorphism;

(C) Prove that for each set X, $\mu^{\mathcal{M}}_X \colon \mathcal{M}\mathcal{M}X \to \mathcal{M}X$ is a (Σ, E)-algebra homomorphism.

By the correspondence of (Σ, E)-algebras and Eilenberg-Moore algebras for $T_{\Sigma,E}$ and (A), we obtain a $T_{\Sigma,E}$-algebra $\alpha^\sharp_X \colon T_{\Sigma,E}\mathcal{M}X \to \mathcal{M}X$ for each set X. These $\alpha^\sharp_X$ give rise to a natural transformation $\alpha^\sharp \colon T_{\Sigma,E}\mathcal{M} \Rightarrow \mathcal{M}$ by (B) and the correspondence of (Σ, E)-homomorphisms and $T_{\Sigma,E}$-homomorphisms. The monad morphism $\iota \colon T_{\Sigma,E} \Rightarrow \mathcal{M}$ is then obtained by (C) and the following theorem[2].

▶ **Theorem 2.** *Let $(\mathcal{M}, \eta^{\mathcal{M}}, \mu^{\mathcal{M}})$ and $(\hat{\mathcal{M}}, \eta^{\hat{\mathcal{M}}}, \mu^{\hat{\mathcal{M}}})$ be two monads. Let $\alpha^\sharp \colon \mathcal{M}\hat{\mathcal{M}} \Rightarrow \hat{\mathcal{M}}$ be a natural transformation such that $\alpha^\sharp_X \colon \mathcal{M}\hat{\mathcal{M}}X \to \hat{\mathcal{M}}X$ is an Eilenberg-Moore algebra for $\mathcal{M}$ and that $\mu^{\hat{\mathcal{M}}}_X \colon \hat{\mathcal{M}}\hat{\mathcal{M}}X \to \hat{\mathcal{M}}X$ is an $\mathcal{M}$-algebra morphism from $(\hat{\mathcal{M}}\hat{\mathcal{M}}X, \alpha^\sharp_{\hat{\mathcal{M}}X})$ to $(\hat{\mathcal{M}}X, \alpha^\sharp_X)$. Then the following is a monad map:*

$$\iota := \mathcal{M} \xRightarrow{\mathcal{M}\eta^{\hat{\mathcal{M}}}} \mathcal{M}\hat{\mathcal{M}} \xRightarrow{\alpha^\sharp} \hat{\mathcal{M}} \,.$$

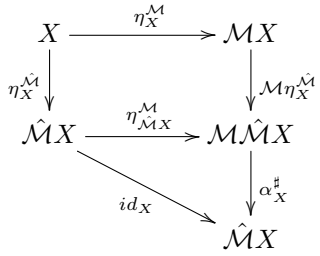
Proof. In order to prove that ι is a monad map, we need to prove that the following two diagrams commute.

$$\begin{array}{ccc} X & \xrightarrow{\eta^{\mathcal{M}}_X} & \mathcal{M}X \\ & \searrow^{\eta^{\hat{\mathcal{M}}}_X} & \downarrow{\scriptstyle \iota_X} \\ & & \hat{\mathcal{M}}X \end{array} \qquad \begin{array}{ccccc} \mathcal{M}\mathcal{M}X & \xrightarrow{\mathcal{M}\iota_X} & \mathcal{M}\hat{\mathcal{M}}X & \xrightarrow{\iota_{\hat{\mathcal{M}}X}} & \hat{\mathcal{M}}\hat{\mathcal{M}}X \\ \downarrow{\scriptstyle \mu^{\mathcal{M}}_X} & & & & \downarrow{\scriptstyle \mu^{\hat{\mathcal{M}}}_X} \\ \mathcal{M}X & & \xrightarrow[\iota_X]{} & & \hat{\mathcal{M}}X \end{array} \tag{6}$$

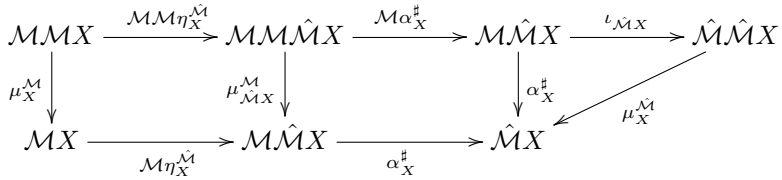
For the diagram on the left, it is enough to recall that $\iota = \alpha^\sharp \circ \mathcal{M}\eta^{\hat{\mathcal{M}}}$ and observe that the following diagram commutes: the top square commutes by naturality of $\eta^{\mathcal{M}}$ and the

[2] This theorem is known, but it is not easy to find an original reference for it. We thank Jurriaan Rot for recalling the theorem and the proof with us.

bottom triangle commutes since $\alpha^\sharp_X$ is an Eilenberg Moore algebra for $\mathcal{M}$.

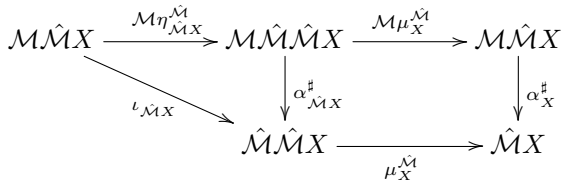


In order to prove the commutation of the diagram on the right in (6), by $\iota = \alpha^\sharp \circ \mathcal{M}\eta^{\hat{\mathcal{M}}}$ it is enough to prove that the following commutes:



The left square commutes by naturality of $\mu^{\mathcal{M}}$. The central square commutes since $\alpha^\sharp_X$ is an Eilenberg-Moore algebra for $\mathcal{M}$. It remains to prove that the right triangle commutes.

First, observe that the diagram below commutes: the left triangle commutes by definition of ι, and the right square commutes by the assumption that $\mu^{\hat{\mathcal{M}}}_X$ is an $\mathcal{M}$-algebra morphism.



This completes the proof as $\mathcal{M}\mu^{\hat{\mathcal{M}}}_X \circ \mathcal{M}\eta^{\hat{\mathcal{M}}}_{\hat{\mathcal{M}}X} = \mathcal{M}(\mu^{\hat{\mathcal{M}}}_X \circ \eta^{\hat{\mathcal{M}}}_{\hat{\mathcal{M}}X}) = \mathcal{M}(id_{\hat{\mathcal{M}}X}) = id_{\mathcal{M}\hat{\mathcal{M}}X}$. ◀

The function $\iota_X : T_{\Sigma,E}X \to \mathcal{M}X$ obtained by the above recipe can be inductively defined for all $x \in X$, $t_1, \ldots, t_n \in T_\Sigma X$ and n-ary operations o in Σ as follows.

$$\iota_X([x]_E) = \eta^{\mathcal{M}}_X(x) \qquad \iota_X([o(t_1, \ldots, t_n)]_E) = \hat{o}_X(\iota_X[t_1]_E, \ldots, \iota_X[t_n]_E). \tag{7}$$

The fact that the functions $\hat{o}_X$ form a (Σ, E)-algebra ensures that ι is a well defined function, namely if $t =_E t'$, then $\iota([t]_E) = \iota([t']_E)$.

We conclude this section by shortly illustrating how to apply the above recipe to the monad for nondeterminism and the one for probability discussed above. To construct a monad map $\iota^N : T_{\Sigma_N, E_N} \Rightarrow \mathcal{P}_{ne}$, we define for all sets X the binary function $\hat{\oplus} : \mathcal{P}_{ne}(X) \times \mathcal{P}_{ne}(X) \to \mathcal{P}_{ne}(X)$ as the union $\cup$. This is associative, commutative and idempotent, so the axioms in E_N are satisfied, or in other words, this forms a semilattice. This corresponds to point (A) of the recipe. It is not difficult to check (B) and (C). The resulting monad map is defined for all sets X as

$$\iota^N_X([x]_{E_N}) = \{x\} \qquad \iota^N_X([t_1 \oplus t_2]_{E_N}) = \iota^N_X([t_1]_{E_N}) \cup \iota^N_X([t_2]_{E_N}).$$

To construct the monad map $\iota^P \colon T_{\Sigma_P, E_P} \Rightarrow \mathcal{D}$, we define for all $p \in (0,1)$ and all sets X the binary function $\hat{+}_p \colon \mathcal{D}(X) \times \mathcal{D}(X) \to \mathcal{D}(X)$ as $d_1 \hat{+}_p d_2 = p\, d_1 + (1-p) d_2$. One can check that the three axioms in E_P are satisfied (distributions form a convex algebra), and that points (B) and (C) of the recipe hold. The resulting monad map is defined for all sets X as

$$\iota^P_X([x]_{E_P}) = \delta_x \qquad \iota^P_X([t_1 +_p t_2]_{E_P}) = p\, \iota^P_X([t_1]_{E_P}) + (1-p)\iota^P_X([t_2]_{E_P}). \tag{8}$$

4 The monad for nondeterminism and probability

In this section, we recall the monad for nondeterminism and probability, its presentation, and we illustrate some interesting properties.

The monad $C \colon \mathbf{Sets} \to \mathbf{Sets}$ maps a set X into CX, namely the set of non-empty, finitely-generated convex subsets of distributions on X (as defined in Section 2). For a function $f \colon X \to Y$, $Cf \colon CX \to CY$ is given by $Cf(S) = \{\mathcal{D}f(d) \mid d \in S\}$. The unit of C is $\eta \colon X \to CX$ given by $\eta(x) = \{\delta_x\}$. The multiplication $\mu \colon CCX \to CX$ of C can be expressed in concrete terms as follows [16]. Given $S \in CCX$,

$$\mu(S) = \bigcup_{\Phi \in S} \Big\{ \sum_{U \in \operatorname{supp} \Phi} \Phi(U) \cdot d \mid d \in U \Big\}.$$

Let Σ be the signature $\Sigma_N \cup \Sigma_P$. Let E be the sets of axioms consisting of E_N, E_p and the following distributivity axiom:

$$(x \oplus y) +_p z \overset{(D)}{=} (x +_p z) \oplus (y +_p z)$$

This theory (Σ, E) is the algebraic theory of *convex semilattices*, introduced in [3].

▶ **Theorem 3.** *(Σ, E) is a presentation of the monad C.*

The above theorem has been proved in [3]. In the remainder of this paper, we will provide an alternative proof of this fact by exploiting the unique base theorem (Theorem 1).

We begin by observing that the assignment $S \mapsto \operatorname{conv}(S)$ gives rise to a natural transformation $\operatorname{conv} \colon \mathcal{P}_{ne}\mathcal{D} \Rightarrow C$ [20, 2]. Theorem 1 provides a way of going backward, from C to $\mathcal{P}_{ne}\mathcal{D}$: we call $\mathrm{UB}_X \colon CX \to \mathcal{P}_{ne}\mathcal{D}X$ the function assigning to each convex subset S its unique base. However such UB_X does not give rise to a natural transformation, in the sense that the diagram on the left in (9) only commutes laxly for arbitrary functions $f \colon X \to Y$.

$$\begin{array}{ccc} CX & \xrightarrow{Cf} & CY \\ {\scriptstyle \mathrm{UB}_X}\downarrow & \swarrow & \downarrow{\scriptstyle \mathrm{UB}_Y} \\ \mathcal{P}_{ne}\mathcal{D}X & \xrightarrow[\mathcal{P}_{ne}\mathcal{D}f]{} & \mathcal{P}_{ne}\mathcal{D}Y \end{array} \qquad \begin{array}{ccc} CX & \xrightarrow{Cf} & CY \\ {\scriptstyle \mathrm{UB}_X}\downarrow & & \uparrow{\scriptstyle \mathrm{conv}_Y} \\ \mathcal{P}_{ne}\mathcal{D}X & \xrightarrow[\mathcal{P}_{ne}\mathcal{D}f]{} & \mathcal{P}_{ne}\mathcal{D}Y \end{array} \tag{9}$$

It holds that $\mathrm{UB}_Y \circ Cf \subseteq \mathcal{P}_{ne}\mathcal{D}f \circ \mathrm{UB}_X$ but not the other way around, as shown by the next example.

▶ **Example 4.** Let $X = \{x, y, z\}$, $Y = \{a, b\}$ and $f \colon X \to Y$ be the function mapping both x and y to a and z to b. Consider the set $S = \{\frac{1}{2}x + \frac{1}{2}y,\ \frac{1}{2}x + \frac{1}{2}z,\ \delta_z\}$: this set is a base since none of its element can be expressed as convex combination of the others. However, the set $\mathcal{P}_{ne}\mathcal{D}f(S) = \{\delta_a,\ \frac{1}{2}a + \frac{1}{2}b,\ \delta_b\}$ is not a base since $\frac{1}{2}a + \frac{1}{2}b$ can be expressed as a linear combination of δ_a and δ_b. Now, by taking the convex set $\operatorname{conv}(S) \in CX$ one can

easily see that $\mathrm{UB}_Y \circ Cf \not\supseteq \mathcal{P}_{ne}\mathcal{D}f \circ \mathrm{UB}_X$. Indeed $\mathcal{P}_{ne}\mathcal{D}f \circ \mathrm{UB}_X(\mathrm{conv}(S)) = \mathcal{P}_{ne}\mathcal{D}f(S) = \{\delta_a, \frac{1}{2}a + \frac{1}{2}b, \delta_b\}$, while $\mathrm{UB}_Y \circ Cf(\mathrm{conv}(S)) = \{\delta_a, \delta_b\}$ since $Cf(\mathrm{conv}(S)) = \mathrm{conv}(\mathcal{D}f(S))$ by Lemma 5 below.

Interestingly enough, while the diagram on the left in (9) does not commute, the diagram on the right in (9) does. This is closely related to Lemma 37 from [3], which provides a slightly different formulation. Below, we illustrate a proof: to simplify the notation of the natural transformations, we avoid to specify the set X whenever it is clear from the context.

▶ **Lemma 5.** *Let $S \in C(X)$ and $f : X \to Y$. Then $Cf(S) = \mathrm{conv}(\{\mathcal{D}f(d) \mid d \in \mathrm{UB}(S)\})$.*

Proof. We prove $Cf(S) \subseteq \mathrm{conv}(\bigcup_{d\in\mathrm{UB}(S)}\{\mathcal{D}f(d)\})$. Let $e \in Cf(S)$. Then $e = \mathcal{D}f(d)$ for some $d \in S$, which implies that d is a convex combination of elements of $\mathrm{UB}(S)$, that is, $d = \sum_i p_i \cdot d_i$ with $d_i \in \mathrm{UB}(S)$ for all i. Hence, $e = \sum_i p_i \cdot \mathcal{D}f(d_i) \in \mathrm{conv}(\bigcup_{d\in\mathrm{UB}(S)}\{\mathcal{D}f(d)\})$.

For the opposite inclusion, let $e \in \mathrm{conv}(\bigcup_{d\in\mathrm{UB}(S)}\{\mathcal{D}f(d)\})$. Hence, $e = \sum_i p_i \cdot \mathcal{D}f(d_i)$ with $d_i \in \mathrm{UB}(S)$ for all i. We have $\sum_i p_i \cdot \mathcal{D}f(d_i) = \mathcal{D}f(\sum_i p_i \cdot d_i)$ and, from $\sum_i p_i \cdot d_i \in S$, we conclude $e \in Cf(S)$. ◀

5 The monad map $\iota : T_{\Sigma,E} \Rightarrow C$

In this section we apply the standard recipe from Section 3.1 to construct a monad map $\iota : T_{\Sigma,E} \Rightarrow C$.

For this aim, we first recall two well-known operations on convex sets: the convex union $\oplus : C(X) \times C(X) \to C(X)$ defined for all $S_1, S_2 \in C(X)$ as

$$S_1 \oplus S_2 = \mathrm{conv}(S_1 \cup S_2)$$

and, for all $p \in (0, 1)$, the Minkowski sum $+_p : C(X) \times C(X) \to C(X)$ defined as

$$S_1 +_p S_2 = \{d \mid d = pd_1 + (1-p)d_2 \text{ for some } d_1 \in S_1 \text{ and } d_2 \in S_2\}.$$

Points (A) and (B) of the recipe hold by the following result from [3, Lemma 38].

▶ **Lemma 6.** *With the above defined operations $(CX, \oplus, +_p)$ is a convex semilattice. Moreover, for a map $f : X \to Y$, the map $Cf : CX \to CY$ is a convex semilattice homomorphism from $(CX, \oplus, +_p)$ to $(CY, \oplus, +_p)$.* ◀

The following lemma proves point (C) explicitly, namely that μ is a (Σ, E)-homomorphism.[3]

▶ **Lemma 7.** *For all $S_1, S_2 \in CC(X)$, it holds that:*

1. $\mu(S_1 \oplus S_2) = \mu(S_1) \oplus \mu(S_2)$
2. $\mu(S_1 +_p S_2) = \mu(S_1) +_p \mu(S_2)$

Proof. Through this proof, we will often use the following key observation: $d \in \mu(S)$ iff

$$\exists \Phi \in S \text{ such that } d = \sum_{U\in\mathrm{supp}(\Phi)} \Phi(U)\cdot f(U) \text{ , for } f : \mathrm{supp}(\Phi) \to \mathcal{D}(X) \text{ such that } f(U) \in U.$$

[3] In [3], we show that $(CX, \oplus, +_p)$ is the free convex semilattice generated by X and then prove that $\mu = id^{\#}_{CX}$, see [3, Lemma 41]. An implicit consequence of this is that μ is the unique homomorphism from the free convex semilattice generated by CX to the free convex semilattice generated by X that extends the identity map on CX.

1. We first prove the inclusion $\mu(S_1) \oplus \mu(S_2) \subseteq \mu(S_1 \oplus S_2)$. As $S_1 \subseteq S_1 \oplus S_2$ we derive that

$$\begin{array}{lll} \mu(S_1) & \stackrel{\text{def}}{=} & \bigcup_{\Phi \in S_1} \{\sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\} \\ & \subseteq & \bigcup_{\Phi \in S_1 \oplus S_2} \{\sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\} \\ & \stackrel{\text{def}}{=} & \mu(S_1 \oplus S_2) \end{array} \tag{10}$$

Symmetrically, by $S_2 \subseteq S_1 \oplus_p S_2$ we have

$$\begin{array}{lll} \mu(S_2) & \stackrel{\text{def}}{=} & \bigcup_{\Phi \in S_2} \{\sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\} \\ & \subseteq & \bigcup_{\Phi \in S_1 \oplus S_2} \{\sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\} \\ & \stackrel{\text{def}}{=} & \mu(S_1 \oplus S_2) \end{array} \tag{11}$$

Hence,

$$\begin{aligned} &\mu(S_1) \oplus \mu(S_2) \\ &\quad = \text{conv}\Big(\bigcup_{\Phi \in S_1} \Big\{ \sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U \Big\} \cup \bigcup_{\Phi \in S_2} \Big\{ \sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U \Big\} \Big) \\ &\quad \subseteq \text{conv}(\mu(S_1 \oplus S_2)) && \text{(by (10), (11))} \\ &\quad = \mu(S_1 \oplus S_2) && \text{(by } \mu(S_1 \oplus S_2) \text{ a convex set)} \end{aligned}$$

We then prove the inclusion $\mu(S_1 \oplus S_2) \subseteq \mu(S_1) \oplus \mu(S_2)$. Take $d \in \mu(S_1 \oplus S_2)$. Then there is a $\Phi \in S_1 \oplus S_2$ such that $d = \sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot f(U)$, with $f : \text{supp}(\Phi) \to \mathcal{D}(X)$ such that $f(U) \in U$. As Φ is a convex combination of distributions in $S_1 \cup S_2$, we have $\Phi = \sum_i p_i \cdot \Phi_i$ with $\Phi_i \in (S_1 \cup S_2)$ for all i. Then for all $x \in X$ we have

$$\begin{aligned} \sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot f(U)(x) &= \sum_{U \in \cup_i \text{supp}(\Phi_i)} \Big((\sum_i p_i \cdot \Phi_i)(U) \cdot f(U)(x) \Big) \\ &= \sum_{U \in \cup_i \text{supp}(\Phi_i)} \Big(\sum_i p_i \cdot \Phi_i(U) \cdot f(U)(x) \Big) \\ &= \sum_i p_i \cdot \Big(\sum_{U \in \cup_i \text{supp}(\Phi_i)} \Phi_i(U) \cdot f(U)(x) \Big) \\ &= \sum_i p_i \cdot \Big(\sum_{U \in \text{supp}(\Phi_i)} \Phi_i(U) \cdot f(U)(x) \Big) \end{aligned}$$

Hence, the result follows as

$$\begin{aligned} d &= \sum_i p_i \cdot \Big(\sum_{U \in \text{supp}(\Phi_i)} \Phi_i(U) \cdot f(U) \Big) \\ &\in \text{conv}\Big(\bigcup_{\Phi \in (S_1 \cup S_2)} \Big\{ \sum_{U \in \text{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U \Big\} \Big) \\ &= \mu(S_1 \oplus S_2). \end{aligned}$$

2. We first prove $\mu(S_1) +_p \mu(S_2) \subseteq \mu(S_1 +_p S_2)$. Let $d \in \mu(S_1) +_p \mu(S_2)$. Then

$$d = \Big(\sum_{U \in \text{supp}(\Phi_1)} \Phi_1(U) \cdot f(U) \Big) +_p \Big(\sum_{U \in \text{supp}(\Phi_2)} \Phi_2(U) \cdot g(U) \Big)$$

with $\Phi_1 \in S_1, \Phi_2 \in S_2$, with $f : \mathrm{supp}(\Phi_1) \to \mathcal{D}(X)$ such that $f(U) \in U$, and with $g : \mathrm{supp}(\Phi_2) \to \mathcal{D}(X)$ such that $g(U) \in U$. For all $x \in X$, we have

$$
\begin{aligned}
d(x) &= \Big(\big(\sum_{U \in \mathrm{supp}(\Phi_1)} \Phi_1(U) \cdot f(U) \big) +_p \big(\sum_{U \in \mathrm{supp}(\Phi_2)} \Phi_2(U) \cdot g(U) \big) \Big)(x) \\
&= \big(\sum_{U \in \mathrm{supp}(\Phi_1)} (p \cdot \Phi_1(U) \cdot f(U)(x)) \big) + \big(\sum_{U \in \mathrm{supp}(\Phi_2)} ((1-p) \cdot \Phi_2(U) \cdot g(U)(x)) \big) \\
&= \big(\sum_{U \in \mathrm{supp}(\Phi_1) \setminus \mathrm{supp}(\Phi_2)} (p \cdot \Phi_1(U) \cdot f(U)(x)) \big) \\
&\quad + \big(\sum_{U \in \mathrm{supp}(\Phi_2) \setminus \mathrm{supp}(\Phi_1)} ((1-p) \cdot \Phi_2(U) \cdot g(U)(x)) \big) \\
&\quad + \Big(\sum_{U \in \mathrm{supp}(\Phi_1) \cap \mathrm{supp}(\Phi_2)} \big((p \cdot \Phi_1(U) \cdot f(U)(x)) + ((1-p) \cdot \Phi_2(U) \cdot g(U)(x)) \big) \Big) \\
&\overset{(*)}{=} \big(\sum_{U \in \mathrm{supp}(\Phi_1) \setminus \mathrm{supp}(\Phi_2)} ((\Phi_1 +_p \Phi_2)(U) \cdot f(U)(x)) \big) \\
&\quad + \big(\sum_{U \in \mathrm{supp}(\Phi_2) \setminus \mathrm{supp}(\Phi_1)} ((\Phi_1 +_p \Phi_2)(U) \cdot g(U)(x)) \big) \\
&\quad + \Big(\sum_{U \in \mathrm{supp}(\Phi_1) \cap \mathrm{supp}(\Phi_2)} \big((\Phi_1 +_p \Phi_2)(U) \cdot (f(U)(x) +_{\frac{p \cdot \Phi_1(U)}{(\Phi_1 +_p \Phi_2)(U)}} g(U)(x)) \big) \Big) \\
&= \sum_{U \in \mathrm{supp}(\Phi_1 +_p \Phi_2)} \big((\Phi_1 +_p \Phi_2)(U) \cdot h(U)(x) \big)
\end{aligned}
$$

where $h : \mathrm{supp}(\Phi_1 +_p \Phi_2) \to \mathcal{D}(X)$ is defined as:

$$
h(U) = \begin{cases} f(U) & \text{if } U \in (\mathrm{supp}(\Phi_1) \setminus \mathrm{supp}(\Phi_2)) \\ g(U) & \text{if } U \in (\mathrm{supp}(\Phi_2) \setminus \mathrm{supp}(\Phi_1)) \\ (f(U) +_{\frac{p \cdot \Phi_1(U)}{(\Phi_1 +_p \Phi_2)(U)}} g(U)) & \text{if } U \in (\mathrm{supp}(\Phi_1) \cap \mathrm{supp}(\Phi_2)) \end{cases}
$$

and the equality $(*)$ holds by $(p_1 \cdot q_1) + (p_2 \cdot q_2) = (p_1 + p_2) \cdot (q_1 +_{\frac{p_1}{p_1+p_2}} q_2)$, $\forall p_1, p_2, q_1, q_2$. Then, observe that for every $U \in \mathrm{supp}(\Phi_1 +_p \Phi_2)$ we have $h(U) \in U$, since every U is a convex set, and thus if U contains $f(U)$ and $g(U)$ then it also contains $f(U) +_q g(U)$, for all q. Thereby, we conclude $d \in \mu(S_1 +_p S_2)$.

We now prove the remaining inclusion, i.e., $\mu(S_1 +_p S_2) \subseteq \mu(S_1) +_p \mu(S_2)$.

Let $\Phi \in S_1 +_p S_2$ and let $d = \sum_{U \in \mathrm{supp}(\Phi)} \Phi(U) \cdot f(U)$, with $f : \mathrm{supp}(\Phi) \to \mathcal{D}(X)$ such that $f(U) \in U$, be an element of $\mu(S_1 +_p S_2)$. Then, $\Phi = \Phi_1 +_p \Phi_2$, with $\Phi_1 \in S_1, \Phi_2 \in S_2$. For every $x \in X$ we have

$$
\begin{aligned}
d(x) &= \sum_{U \in \mathrm{supp}(\Phi_1) \cup \mathrm{supp}(\Phi_2)} \Big((\Phi_1 +_p \Phi_2)(U) \cdot f(U)(x) \Big) \\
&= \sum_{U \in \mathrm{supp}(\Phi_1) \cup \mathrm{supp}(\Phi_2)} \Big((p \cdot \Phi_1(U) \cdot f(U)(x)) + ((1-p) \cdot \Phi_2(U) \cdot f(U)(x)) \Big) \\
&= \Big(\sum_{U \in \mathrm{supp}(\Phi_1)} p \cdot \Phi_1(U) \cdot f(U)(x) \Big) + \Big(\sum_{U \in \mathrm{supp}(\Phi_2)} (1-p) \cdot \Phi_2(U) \cdot f(U)(x) \Big) \\
&= \Big(\sum_{U \in \mathrm{supp}(\Phi_1)} \Phi_1(U) \cdot f(U)(x) \Big) +_p \Big(\sum_{U \in \mathrm{supp}(\Phi_2)} \Phi_2(U) \cdot f(U)(x) \Big)
\end{aligned}
$$

which implies $d \in \mu(S_1) +_p \mu(S_2)$. ◀

By applying the recipe from Section 3.1, we obtain from Lemmas 6 and 7 a monad map.

▶ **Proposition 8.** *The natural transformation* $\iota\colon T_{\Sigma,E} \Rightarrow C$ *is a monad map, defined as:*

$$\iota([x]_E) = \{\delta_x\} \quad \iota([t_1 \oplus t_2]_E) = \iota([t_1]_E) \oplus \iota([t_2]_E) \quad \iota([t_1 +_p t_2]_E) = \iota([t_1]_E) +_p \iota([t_2]_E)$$

Lemma 7, together with the existence of unique bases, also allows us to derive a useful characterization of the multiplication μ of the monad C.

▶ **Lemma 9.** *For* $S \in CCX$,

$$\mu(S) = \operatorname{conv}\Big(\bigcup_{\Phi \in \mathrm{UB}(S)} \Big\{ \sum_{U \in \operatorname{supp} \Phi} \Phi(U) \cdot d \mid d \in \mathrm{UB}(U) \Big\} \Big).$$

Proof. We have $S = \operatorname{conv}(\bigcup_{\Phi \in \mathrm{UB}(S)} \{\Phi\})$ which means that S is a convex union of the sets $\{\Phi\}$, for $\Phi \in \mathrm{UB}(S)$. Then by Lemma 7 we derive $\mu(S) = \operatorname{conv}(\bigcup_{\Phi \in \mathrm{UB}(S)} \mu\{\Phi\})$. By definition, $\mu\{\Phi\} = \{\sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\}$, hence

$$\mu(S) = \operatorname{conv}\Big(\bigcup_{\Phi \in \mathrm{UB}(S)} \Big\{ \sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U \Big\} \Big). \tag{12}$$

Observe that the Minkowski sum operation, which is equivalently defined on arbitrary (i.e., not necessarily convex) sets of distributions, enjoys the following property:

$$\text{for any sets of distributions } X, Y,\ \operatorname{conv}(X) +_p \operatorname{conv}(Y) = \operatorname{conv}(X +_p Y). \tag{13}$$

Indeed, $X +_p Y \subseteq \operatorname{conv}(X) +_p \operatorname{conv}(Y)$, and as the Minkowski sum of convex sets is convex we have $\operatorname{conv}(X +_p Y) \subseteq \operatorname{conv}(\operatorname{conv}(X) +_p \operatorname{conv}(Y)) = \operatorname{conv}(X) +_p \operatorname{conv}(Y)$. For the other direction, take $p(\sum_i p_i x_i) + (1-p)(\sum_j q_j y_j) \in \operatorname{conv}(X) +_p \operatorname{conv}(Y)$. We have:

$$p(\sum_i p_i x_i) + (1-p)(\sum_j q_j y_j) = p(\sum_{i,j} (p_i q_j) x_i) + (1-p)(\sum_{i,j} (p_i q_j) y_j) = \sum_{i,j} (p_i q_j)(p x_i + (1-p) y_j)$$

which is then an element of $\operatorname{conv}(X +_p Y)$. This shows (13).

For every Φ, the set $\{\sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U\}$ is a Minkowski sum over the elements U of $\operatorname{supp}(\Phi)$, which are themselves convex sets satisfying $U = \operatorname{conv}(\mathrm{UB}(U))$. Then by (13) we derive:

$$\Big\{ \sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in U \Big\} = \operatorname{conv}\Big(\Big\{ \sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in \mathrm{UB}(U) \Big\} \Big). \tag{14}$$

By (12) and (14) it holds:

$$\mu(S) = \operatorname{conv}\left(\bigcup_{\Phi \in \mathrm{UB}(S)} \operatorname{conv}\Big(\Big\{ \sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in \mathrm{UB}(U) \Big\} \Big) \right).$$

Then, by using the property that $\operatorname{conv}(\operatorname{conv}(X) \cup Y) = \operatorname{conv}(X \cup Y)$ for any sets of distributions X, Y (as shown in the proof of [3, Lemma 38]), we conclude that the latter is equal to

$$\operatorname{conv}\Big(\bigcup_{\Phi \in \mathrm{UB}(S)} \Big\{ \sum_{U \in \operatorname{supp}(\Phi)} \Phi(U) \cdot d \mid d \in \mathrm{UB}(U) \Big\} \Big).$$

◀

6 Proving the isomorphism

In the previous section we have constructed a monad map $\iota\colon T_{\Sigma,E} \Rightarrow C$ (Proposition 8). In this section, we prove that it is an isomorphism by exploting Theorem 1.

We start with a simple observation: for each set X, there is a trivial injection $i_X\colon T_{\Sigma_P}(X) \to T_\Sigma(X)$. A term in T_Σ is said to be a *purely probabilistic term* (p-term, for short) if and only if it lays in the image of i. We overload the notation and also denote with i its extension to equivalence classes $i_X\colon T_{\Sigma_P,E_P}(X) \to T_{\Sigma,E}(X)$, which is well defined as $E_P \subseteq E$.

▶ **Lemma 10.** *Let $\{-\}_X\colon \mathcal{D}(X) \to C(X)$ be the function mapping every distribution d into the convex set $\{d\}$ and let $\iota^P\colon T_{\Sigma_P,E_P} \Rightarrow \mathcal{D}$ be the monad map from (8). The following diagram commutes.*

$$\begin{array}{ccc} T_{\Sigma_P,E_P}X & \xrightarrow{i_X} & T_{\Sigma,E}X \\ {\scriptstyle \iota^P_X}\downarrow & & \downarrow{\scriptstyle \iota_X} \\ \mathcal{D}X & \xrightarrow[\{-\}_X]{} & CX \end{array}$$

Proof. We prove by induction that $\{\iota^P_X([t]_{E_P})\}_X = \iota_X(i_X([t]_{E_P}))$ for all $t \in T_{\Sigma_P}$. If $t = x \in X$, then $\{\iota^P_X([x]_{E_P})\}_X = \{\delta_x\} = \iota_X([x]_E) = \iota_X(i_X([t]_{E_P}))$. If $t = t_1 +_p t_2$, then

$$\begin{aligned} \{\iota^P_X([t_1 +_p t_2]_{E_P})\}_X &= \{p \cdot \iota^P_X([t_1]_{E_P}) + (1-p) \cdot \iota^P_X([t_1]_{E_P})\} \\ &= \{\iota^P_X([t_1]_{E_P})\} +_p \{\iota^P_X([t_2]_{E_P})\} \\ &= \iota_X(i_X([t_1]_{E_P})) +_p \iota_X(i_X([t_2]_{E_P})) \\ &= \iota_X([t_1]_E) +_p \iota_X([t_2]_E) \\ &= \iota_X([t_1 +_p t_2]_E) \\ &= \iota_X(i_X([t_1 +_p t_2]_{E_P})). \end{aligned}$$

◀

Recall that the monad map $\iota^P\colon T_{\Sigma_P,E_P} \Rightarrow \mathcal{D}$ defined in (8) is an isomorphism. We call $\kappa^P\colon \mathcal{D} \Rightarrow T_{\Sigma_P,E_P}$ its inverse. By exploiting κ^P and Theorem 1, it is easy to define a function $\kappa_X\colon C(X) \to T_{\Sigma,E}(X)$ as follows: for $S \in C(X)$ with base $\{d_1,\dots,d_n\}$

$$\kappa_X(S) = [\, i(\kappa^P(d_1)) \oplus \dots \oplus i(\kappa^P(d_n)) \,]_E. \tag{15}$$

▶ **Proposition 11.** $\iota \circ \kappa = id_C$.

Proof. Let $S \in C(X)$ be a convex set with base $\{d_1,\dots,d_n\}$. By definition of κ and ι,

$$\iota(\kappa(S)) = \iota([i(\kappa^P(d_1))]_E) \oplus \dots \oplus \iota([i(\kappa^P(d_n))]_E).$$

By Lemma 10, $\iota(\kappa(S)) = \{d_1\} \oplus \dots \oplus \{d_n\}$ which is exactly S. ◀

▶ Remark 12. Proposition 11 and Lemma 10 entail that $i_X\colon T_{\Sigma_P,E_P}(X) \to T_{\Sigma,E}(X)$ is injective. Hence, two p-terms are equal in E if and only if they are also equal in E_P.

We are now left to prove that $\kappa \circ \iota = id_{T_{\Sigma,E}}$. This means that any term t is in the equivalence class of $\kappa \circ \iota([t]_E)$, which by definition of κ is $[i(\kappa^P(d_1)) \oplus \dots \oplus i(\kappa^P(d_n))]_E$ where $\{d_1,\dots,d_n\}$ is the base of $\iota([t]_E)$.

The first step consists in showing that every term is equivalent, modulo E, with a term of a certain shape: a term $t \in T_\Sigma(X)$ is said to be in *nondeterministic-probablistic form*, *n-p*

form for short, if there exists $t_1, \dots, t_n \in T_{\Sigma_P}(X)$ such that $t = i(t_1) \oplus \dots \oplus i(t_n)$. This can be thought of as an analogous of the disjunctive-conjunctive form that is commonly used in propositional logic.

▶ **Example 13.** The term $(x \oplus y) +_{\frac{1}{2}} (y +_{\frac{1}{3}} z)$ is not in n-p form, since $x \oplus y$ occurs inside $+_{\frac{1}{2}}$. However, by using the distributivity axiom (D), we have that $(x \oplus y) +_{\frac{1}{2}} (y +_{\frac{1}{3}} z) =_E (x +_{\frac{1}{2}} (y +_{\frac{1}{3}} z)) \oplus (y +_{\frac{1}{2}} (y +_{\frac{1}{3}} z))$ which is in n-p form.

The following proposition ensures that every term is equivalent to one in n-p form.

▶ **Proposition 14.** *For all $t \in T_\Sigma(X)$, there exists t' in n-p form such that $t =_E t'$.*

Proof. Intuitively, by virtue of the axiom (D) all the occurrences of $+_p$ can be pushed inside some $\oplus$. This can be proved formally by means of the following term rewriting system.

$$(t_1 \oplus t_2) +_p t_3 \rightsquigarrow (t_1 +_p t_3) \oplus (t_2 +_p t_3) \qquad t_1 +_p (t_2 \oplus t_3) \rightsquigarrow (t_1 +_p t_2) \oplus (t_1 +_p t_3)$$

If $t \in T_\Sigma(X)$ rewrites to $t' \in T_\Sigma(X)$, then $t =_E t'$ since the left rule is just the axiom (D), while the right can be derived using (C_p), (D) and (C_p) again. Using standard term rewriting techniques from [6] we can prove that the rewriting system terminates:

(1) Define the partial order $+_p > \oplus$ on Σ;
(2) Observe that the generated recursive path ordering on $T_\Sigma(X)$ is a simplification ordering (see e.g., Example A in Section 5 of [6]);
(3) Conclude by the First Termination Theorem.

Finally, we observe that a term t is in n-p form iff $t \not\rightsquigarrow$: Indeed, if t is in n-p form then there is no redex for the two rules above. On the other hand, if t is not in n-p form, then some $\oplus$ should occur inside a $+_p$ and then one of the rules applies.

Therefore, each term t can be rewritten into an E-equivalent term t' in n-p form. ◀

Given a term $t' \in T_\Sigma(X)$ in n-p form and $t_1, \dots, t_n \in T_{\Sigma_P}(X)$ such that $t' = i(t_1) \oplus \dots \oplus i(t_n)$, one would like $\{\iota^P([t_1]_{E_P}), \dots, \iota^P([t_n]_{E_P})\}$ to be the base for $\iota([t']_E)$. But this is not always the case since some $\iota^P([t_i]_{E_P})$ can be a convex combination of the other $\iota^P([t_j]_{E_P})$.

▶ **Example 15.** The term $(x +_{\frac{1}{2}} y) \oplus (x +_{\frac{2}{3}} (x \oplus y))$ is not in n-p form. By applying the rewriting procedure in the proof of Proposition 14 one obtains: $(x +_{\frac{1}{2}} y) \oplus (x +_{\frac{2}{3}} (x \oplus y)) =_E (x +_{\frac{1}{2}} y) \oplus (x +_{\frac{2}{3}} x) \oplus (x +_{\frac{2}{3}} y)$. Observe that this is equivalent to $(x +_{\frac{1}{2}} y) \oplus x \oplus (x +_{\frac{2}{3}} y)$. The convex set $\iota\big([(x +_{\frac{1}{2}} y) \oplus x \oplus (x +_{\frac{2}{3}} y)]_E\big)$ has base $\{\iota^P([x +_{\frac{1}{2}} y]_{E_P}), \iota_P([x]_{E_P})\} = \{\frac{1}{2}x + \frac{1}{2}y, \delta_x\}$. Indeed the distribution $\iota^P([x +_{\frac{2}{3}} y]_{E_P}) = \frac{2}{3}x + \frac{1}{3}y$ is a convex combination of $\{\frac{1}{2}x + \frac{1}{2}y, \delta_x\}$ as $\frac{2}{3}x + \frac{1}{3}y = \frac{2}{3}(\frac{1}{2}x + \frac{1}{2}y) + \frac{1}{3}x$.

The next two lemmas are necessary to show that, using the axioms in E, we can remove from t' those summands $i(t_i)$ such that $\iota^P([t_i]_{E_P})$ is a convex combination of the other $\iota^P([t_j]_{E_P})$. The first lemma is a well known observation (see e.g. [23, 33]), but we report its instructive proof; the second lemma follows easily from the first one and properties of convex algebras. We defer its proof to the Appendix.

▶ **Lemma 16** (Convexity law). *For all terms $t_1, t_2 \in T_\Sigma(X)$, for all $p \in (0, 1)$,*

$$t_1 \oplus t_2 =_E t_1 \oplus t_2 \oplus (t_1 +_p t_2).$$

Proof. We first prove that

$$\begin{aligned} t_1 \oplus t_2 &\overset{(I_p)}{=} (t_1 \oplus t_2) +_p (t_1 \oplus t_2) \\ &\overset{(D)}{=} ((t_1 \oplus t_2) +_p t_1) \oplus ((t_1 \oplus t_2) +_p t_2) \\ &\overset{(D)}{=} ((t_1 +_p t_1) \oplus (t_2 +_p t_1)) \oplus ((t_1 +_p t_2) \oplus (t_2 +_p t_2)) \\ &\overset{(I_p)}{=} t_1 \oplus (t_2 +_p t_1) \oplus (t_1 +_p t_2) \oplus t_2 \end{aligned}$$

Then, by applying first this equality and then idempotency, we derive the result:

$$\begin{aligned} t_1 \oplus t_2 \oplus (t_1 +_p t_2) &= t_1 \oplus (t_2 +_p t_1) \oplus (t_1 +_p t_2) \oplus t_2 \oplus (t_1 +_p t_2) \\ &\overset{(I_p)}{=} t_1 \oplus (t_2 +_p t_1) \oplus (t_1 +_p t_2) \oplus t_2 \end{aligned}$$

◀

▶ **Lemma 17.** *Let* $t, t_1, \ldots, t_n \in T_{\Sigma_P}(X)$ *such that* $\iota^P([t]_{E_P}) \in \mathrm{conv}\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_n]_{E_P})\}$. *Then*

$$i(t_1) \oplus \ldots \oplus i(t_n) =_E i(t_1) \oplus \ldots \oplus i(t_n) \oplus i(t).$$

▶ **Proposition 18.** *For all terms* $t \in T_\Sigma(X)$, *there exist* $t_1, \ldots, t_n \in T_{\Sigma_P}$ *such that*

$$t =_E i(t_1) \oplus \ldots \oplus i(t_n)$$

and $\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_n]_{E_P})\}$ *is the base of* $\iota([t]_E)$.

Proof. By Proposition 14, there exists a $t' \in T_\Sigma(X)$ in n-p form such that $t =_E t'$. Take $t'_1, \ldots, t'_m \in T_{\Sigma_P}$ such that $t' = i(t_1) \oplus \ldots \oplus i(t_m)$. By definition of ι, $\iota([t]_E) = \iota(i([t_1]_{E_P})) \oplus \cdots \oplus \iota(i([t_m]_{E_P}))$ which by Lemma 10 is $\{\iota^P([t_1]_{E_P})\} \oplus \cdots \oplus \{\iota^P([t_m]_{E_P})\}$. By definition of $\oplus$, this is just $\mathrm{conv}\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_m]_{E_P})\}$. Therefore, to conclude that $\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_m]_{E_P})\}$ is the base of $\iota([t]_E)$ we only need to show that none of the $\iota^P([t_i]_{E_P})$ is in the convex combination of the others $\iota^P([t_j]_{E_P})$. This is not true in general, but thanks to Lemma 17 all such t_i can be removed, while preserving E-equivalence. To be more precise, by associativity and commutativity of $\oplus$, we can assume that $\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_n]_{E_P})$ form the base, while $\iota^P([t_{n+1}]_P), \ldots, \iota^P([t_m]_{E_P})$ are in $\mathrm{conv}\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t'_n]_{E_P})\}$. Then, by repeating $(m-n)$-times Lemma 17, we conclude that $t' =_E i(t_1) \oplus \ldots \oplus i(t_n)$. ◀

▶ **Proposition 19.** $\kappa \circ \iota = id_{T_{\Sigma,E}}$.

Proof. We need to prove that for all terms $t \in T_\Sigma(X)$, $[t]_E = \kappa \circ \iota([t]_E)$. By Proposition 18, there exists $t_1, \ldots, t_n \in T_{\Sigma_P}(X)$ such that

$$t =_E i(t_1) \oplus \ldots \oplus i(t_n)$$

and $\{\iota^P([t_1]_{E_P}), \ldots, \iota^P([t_n]_{E_P})\}$ is the base for $\iota([t]_E)$.
By definition of κ, $\kappa(\iota([t]_E))$ is exactly $[i(\kappa^P \circ \iota^P[t_1]_{E_P}) \oplus \ldots \oplus i(\kappa^P \circ \iota^P[t_n]_{E_P})]_E = [t]_E$. ◀

This is enough to conclude the proof of Theorem 3. Indeed we have that $\iota \colon T_{\Sigma,E} \Rightarrow C$ is a monad map and that, by Propositions 11 and 19, it is an isomorphism.

References

1 Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

2 Filippo Bonchi, Alexandra Silva, and Ana Sokolova. The Power of Convex Algebras. In *CONCUR 2017*, volume 85, pages 23:1–23:18. LIPIcs, 2017. doi:10.4230/LIPIcs.CONCUR.2017.23.

3 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The theory of traces for systems with nondeterminism and probability. Extended version of paper in Proc.LICS'19, 2019. URL: http://arxiv.org/abs/1808.00923v3.

4 Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Equivalence relations in fully and partially observable markov decision processes. In *IJCAI*, pages 1653–1658, 2009.

5 Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *Proc. CAV 2017*, volume 10427 of *LNCS*, pages 592–600, 2017.

6 Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical computer science*, 17(3):279–301, 1982.

7 Ernst-Erich Doberkat. Eilenberg-Moore algebras for stochastic relations. *Inform. and Comput.*, 204(12):1756–1781, 2006. doi:10.1016/j.ic.2006.09.001.

8 Ernst-Erich Doberkat. Erratum and addendum: Eilenberg-Moore algebras for stochastic relations [mr2277336]. *Inform. and Comput.*, 206(12):1476–1484, 2008. doi:10.1016/j.ic.2008.08.002.

9 Jean Goubault-Larrecq. Prevision domains and convex powercones. In *FOSSACS 2008*, pages 318–333. LNCS 4962, 2008. doi:10.1007/978-3-540-78499-9_23.

10 Alexandre Goy and Daniela Petrisan. Combining probabilistic and non-deterministic choice via weak distributive laws. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 454–464. ACM, 2020. doi:10.1145/3373718.3394795.

11 Hans A Hansson. Time and probability in formal design of distributed systems. *PhD thesis, Uppsala University*, 1991.

12 Holger Hermanns, Jan Krcál, and Jan Kretínský. Probabilistic bisimulation: Naturally on distributions. In *Proc. CONCUR'14*, volume 8704 of *LNCS*, pages 249–265, 2014.

13 Holger Hermanns, Augusto Parma, Roberto Segala, Björn Wachter, and Lijun Zhang. Probabilistic logical characterization. *Information and Computation*, 209(2):154–172, 2011.

14 Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. *CoRR*, abs/1701.02547, 2017. URL: http://arxiv.org/abs/1701.02547.

15 B. Jacobs. Convexity, duality and effects. In *Theoretical computer science*, volume 323 of *IFIP Adv. Inf. Commun. Technol.*, pages 1–19. Springer, Berlin, 2010. doi:10.1007/978-3-642-15240-5_1.

16 Bart Jacobs. Coalgebraic trace semantics for combined possibilitistic and probabilistic systems. *Electr. Notes Theor. Comput. Sci.*, 203(5):131–152, 2008.

17 Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 1998.

18 Klaus Keimel and Gordon D. Plotkin. Mixed powerdomains for probability and nondeterminism. *Logical Methods in Computer Science*, 13(1), 2017. doi:10.23638/LMCS-13(1:2)2017.

19 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *Computer Performance Evaluation / TOOLS*, pages 200–204. LNCS 2324, 2002.

20 Matteo Mio. Upper-expectation bisimilarity and łukasiewicz μ-calculus. In *Proc. FOSSACS'14*, volume 8412 of *LNCS*, pages 335–350, 2014.

21 Matteo Mio, Ralph Sarkis, and Valeria Vignudelli. Combining nondeterminism, probability, and termination: Equational and metric reasoning. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–14. IEEE, 2021. doi:10.1109/LICS52264.2021.9470717.

22 Matteo Mio and Valeria Vignudelli. Monads and quantitative equational theories for nondeterminism and probability. In Igor Konnov and Laura Kovács, editors, *31st International Conference on Concurrency Theory, CONCUR 2020, September 1-4, 2020, Vienna, Austria (Virtual Conference)*, volume 171 of *LIPIcs*, pages 28:1–28:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.CONCUR.2020.28.

23 M. Mislove, J. Ouaknine, and J. Worrell. Axioms for probability and nondeterminism. In *Proc. of the 10th Int. Workshop on Expressiveness in Concurrency (EXPRESS 2003)*, volume 96 of *ENTCS*, pages 7–28. Elsevier, 2003.

24 Michael W. Mislove. Nondeterminism and probabilistic choice: Obeying the laws. In *CONCUR 2000*, pages 350–364. LNCS 1877, 2000. doi:10.1007/3-540-44618-4_26.

25 Walter Rudin. *Functional Analysis*. McGraw-Hill, 1991.

26 Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

27 Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

28 Zbigniew Semadeni. *Monads and their Eilenberg-Moore algebras in functional analysis*. Queen's University, Kingston, Ont., 1973. Queen's Papers in Pure and Applied Mathematics, No. 33.

29 Sam Staton, Hongseok Yang, Frank Wood, Chris Heunen, and Ohad Kammar. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 525–534, 2016. doi:10.1145/2933575.2935313.

30 T. Świrszcz. Monadic functors and convexity. *Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys.*, 22:39–42, 1974.

31 Regina Tix, Klaus Keimel, and Gordon D. Plotkin. Semantic domains for combining probability and non-determinism. *ENTCS*, 222:3–99, 2009. doi:10.1016/j.entcs.2009.01.002.

32 R. Tyllerr. *Convex Analysis*. Princeton University Press, 1972.

33 D. Varacca. *Probability, Nondeterminism and Concurrency: Two Denotational Models for Probabilistic Computation*. PhD thesis, Univ. Aarhus, 2003. BRICS Dissertation Series, DS-03-14.

34 D. Varacca and G. Winskel. Distributing probabililty over nondeterminism. *MSCS*, 16(1):87–113, 2006.

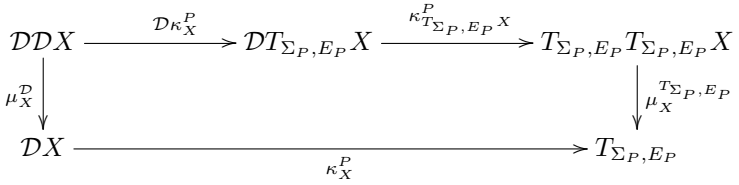
35 Moshe Y Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Foundations of Computer Science, 1985., 26th Annual Symposium on*, pages 327–338. IEEE, 1985.

A Proof of Lemma 17

▶ **Lemma 20.** *Let* $t, t_1, \dots, t_n \in T_{\Sigma_P}(X)$ *such that* $\iota^P([t]_{E_P}) \in \mathrm{conv}\{\iota^P([t_1]_{E_P}), \dots, \iota^P([t_n]_{E_P})\}$. *Then there exist* $p_1 \dots p_{n-1} \in (0,1)$ *such that* $t =_{E_P} (\dots(t_1 +_{p_1} t_2) +_{p_2} \dots) +_{p_{n-1}} t_n$.

Proof. If $\iota^P([t]_{E_P}) \in \mathrm{conv}\{\iota^P([t_1]_{E_P}), \dots, \iota([t_n]_{E_P})\}$, then $\iota^P([t]_{E_P}) = \mu^{\mathcal{D}}\big(\sum_i q_i\,(\iota^P([t_i]_{E_P}))\big)$ for some $\sum_i q_i = 1$. Since ι^P is a monad map, its inverse $\kappa^D \colon \mathcal{D} \Rightarrow T_{\Sigma_P, E_P}$ is also a monad map and in particular it makes the following diagram

commute.



Therefore, we have that

$$\begin{aligned}
[t]_{E_P} &= \kappa^P \circ \iota^P([t]_{E_P}) \\
&= \kappa^P \circ \mu^{\mathcal{D}}\Big(\sum_i q_i \, (\iota^P([t_i]_{E_P}))\Big) \\
&= \mu^{T_{\Sigma_P,E_P}} \circ \kappa^P_{T_{\Sigma_P,E_P}} \circ \mathcal{D}\kappa^P\Big(\sum_i q_i \, (\iota^P([t_i]_{E_P}))\Big) \\
&= \mu^{T_{\Sigma_P,E_P}} \circ \kappa^P_{T_{\Sigma_P,E_P}}\Big(\sum_i q_i \, (\kappa^P \circ \iota^P([t_i]_{E_P}))\Big) \\
&= \mu^{T_{\Sigma_P,E_P}} \circ \kappa^P_{T_{\Sigma_P,E_P}}\Big(\sum_i q_i \, [t_i]_{E_P}\Big)
\end{aligned}$$

Observe that $\sum_i q_i \, [t_i]_{E_P} \in \mathcal{D}T_{\Sigma_p,E_P}(X)$ and that $\kappa^P_{T_{\Sigma_p,E_P}X}$ maps it into an element of $T_{\Sigma_p,E_P}T_{\Sigma_p,E_P}(X)$, namely a term obtained by the operations $+_p$ and the constants $[t_i]_{E_P}$. Using the axioms in E_P any such term can always be written as $(\dots([t_1]_{E_P} +_{p_1} [t_2]_{E_P}) +_{p_2} \dots) +_{p_{n-1}} [t_n]_{E_P}$ for some $p_i \in (0,1)$. Then, the application of $\mu^{T_{\Sigma_P,E_P}}$ to $[(\dots([t_1]_{E_P} +_{p_1} [t_2]_{E_P}) +_{p_2} \dots) +_{p_{n-1}} [t_n]_{E_P}]_{E_P}$ gives $[(\dots(t_1 +_{p_1} t_2) +_{p_2} \dots) +_{p_{n-1}} t_n]_{E_P}$. Thus $t =_{E_P} (\dots(t_1 +_{p_1} t_2) +_{p_2} \dots) +_{p_{n-1}} t_n$. ◀

Proof of Lemma 17. By Lemma 20, we take $p_1, \dots, p_{n-1}$ such that

$$t =_{E_P} (\dots(t_1 +_{p_1} t_2) +_{p_2} \dots) +_{p_{n-1}} t_n. \tag{16}$$

By Lemma 16, $i(t_1) \oplus \dots \oplus i(t_n)$ is E-equivalent to $i(t_1) \oplus \dots \oplus i(t_n) \oplus i(t_1 +_{p_1} t_2)$. By applying Lemma 16 again, one obtains $i(t_1) \oplus \dots \oplus i(t_n) \oplus i(t_1 +_{p_1} t_2) \oplus i((t_1 +_{p_1} t_2) +_{p_2} t_3)$. We can then remove $i(t_1 +_{p_1} t_2)$ using Lemma 16, to obtain

$$i(t_1) \oplus \dots \oplus i(t_n) \oplus i((t_1 +_{p_1} t_2) +_{p_2} t_3).$$

By iterating this procedure, one obtains

$$i(t_1) \oplus \dots \oplus i(t_n) \oplus i((\dots(t_1 +_{p_1} t_2) +_{p_2} \dots) +_{p_{n-1}} t_n)$$

which, by (16), is $i(t_1) \oplus \dots \oplus i(t_n) \oplus i(t)$. ◀

Closure Hyperdoctrines

Davide Castelnovo
Department of Mathematics, Computer Science and Physics, University of Udine, Italy

Marino Miculan
Department of Mathematics, Computer Science and Physics, University of Udine, Italy

Abstract

(Pre)closure spaces are a generalization of topological spaces covering also the notion of neighbourhood in discrete structures, widely used to model and reason about spatial aspects of distributed systems.

In this paper we present an abstract theoretical framework for the systematic investigation of the logical aspects of closure spaces. To this end, we introduce the notion of *closure (hyper)doctrines*, i.e. doctrines endowed with inflationary operators (and subject to suitable conditions). The generality and effectiveness of this concept is witnessed by many examples arising naturally from topological spaces, fuzzy sets, algebraic structures, coalgebras, and covering at once also known cases such as Kripke frames and probabilistic frames (i.e., Markov chains). By leveraging general categorical constructions, we provide axiomatisations and sound and complete semantics for various fragments of logics for closure operators. Hence, closure hyperdoctrines are useful both for refining and improving the theory of existing spatial logics, and for the definition of new spatial logics for new applications.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics

Keywords and phrases categorical logic, topological semantics, closure operators, spatial logic

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.12

Related Version *Full Version*: `https://arxiv.org/abs/2007.04213`

Funding *Marino Miculan*: Supported by the Italian MIUR project PRIN 2017FTXR7S *IT MATTERS (Methods and Tools for Trustworthy Smart Systems)*.

1 Introduction

Recently, much attention has been devoted in Computer Science to systems distributed in physical space; a typical example is provided by the so called *collective adaptive systems*, such as drone swarms, sensor networks, autonomous vehicles, etc. This begs the question of how to model and reason formally about spatial aspects of distributed systems. To this end, several researchers have advocated the use of *spatial logics*, i.e. modal logics whose modalities are interpreted using topological concepts of neighbourhood and connectivity.[1]

In fact, the interpretation of modal logics in topological spaces goes back to Tarski; we refer to [1] for a comprehensive discussion of variants and computability and complexity aspects. More recently, Ciancia *et al.* [10, 11] extended this approach to *preclosure spaces*, also called *Čech closure spaces*, which generalise topological spaces by not requiring idempotence of closure operator. This generalization unifies the notions of neighbourhood arising from topological spaces and from *quasi-discrete closure spaces*, like those induced by graphs and images. Building on this generalization, [10] introduced *Spatial Logic for Closure Spaces* (SLCS), a modal logic for the specification and verification on spatial concepts over preclosure spaces. This logic features a *closure* modality and a spatial *until* modality: intuitively $\phi \mathcal{U} \psi$ holds in an area where ϕ holds and it is not possible to "escape" from it unless passing

[1] Not to be confused with spatial logics for reasoning on the structure of agents, such as the Ambient Logic [8] or the Brane Logic [36].

through an area where ψ holds. SLCS has been proved to be quite effective and expressive, as it has been applied to reachability problems, vehicular movement, digital image analysis (e.g., street maps, radiological images [5]), etc. The model checking problem for this logic over finite quasi-discrete structures is decidable in linear time [10].

Despite these results, an axiomatisation for SLCS is still missing; moreover, it is not obvious how to extend this logic to other spaces with closure operators, such as probabilistic automata (e.g. Markov chains). In fact, the main point is that we miss an abstract theoretical framework for investigating the logical aspects of (pre)closure spaces. Such a framework would be the basis for analysing spatial logics like SLCS, but also for developing further extensions and applications thereof.

In this paper, we aim to build such a framework. To this end, we introduce the new notion of *closure (hyper)doctrine* as the theoretical basis for studying the logical aspects of closure spaces. Doctrines were introduced by Lawvere [30] as a general way for endowing (the objects of) a category with logical notions from a suitable 2-category $\mathbf{E}$, which can be the category of Heyting algebras in the case of intuitionistic logic, of Boolean algebras in the case of classical logic, etc.. Along this line, in order to capture the logical aspects of closure spaces we introduce the notion of *closure operators* on doctrines, that is, families of inflationary morphisms over objects of $\mathbf{E}$ (subject to suitable conditions); a closure (hyper)doctrine is a (hyper)doctrine endowed with a closure operator. These structures arise from many common situations: we provide many examples ranging from topology to algebraic structures, from coalgebras to fuzzy sets. These examples cover the usual cases from literature (e.g., graphs, quasi-discrete spaces, (pre)topological spaces) but include also new settings, such as categories of coalgebras and probabilistic frames (i.e., Markov chains). Then, leveraging general machinery from categorical logic, we introduce a first order logic for closure spaces for which we provide an axiomatisation and a sound and complete categorical semantics. The propositional fragment corresponds to the SLCS from [10].

Overall, the importance of this work is twofold: on one hand, closure hyperdoctrines are useful for analysing and improving the theory of existing spatial logics; in particular, the proposed axiomatisation can enable both new proof methodologies and minimisation techniques. On the other, closure hyperdoctrines are useful for the definition of new logics when we have to deal with closure operators, connectivity, surroundedness, reachability, etc.

Synopsis. The paper is organized as follows. In Section 2 we recall (hyper)doctrines and introduce the key notion of closure doctrine. Many examples of closure doctrines are provided in Section 3. In Section 4 we introduce *logics for closure operators*, together with a sound and complete semantics in closure hyperdoctrines. Conclusions and directions for future work are in Section 5. Longer proofs are in Appendix A.

2 Closure (hyper)doctrines

2.1 Kinds of doctrines

In this section we recall the notion of elementary hyperdoctrine, due to Lawvere [30, 31]. The development of semantics of logics in this context or in the equivalent fibrational context is well established; we refer the reader to, e.g., [23, 35, 38].

▶ **Definition 2.1** ((Existential) Doctrine, Hyperdoctrine [28, 34, 37]). *A* primary doctrine *or simply a* doctrine *on a category* $\mathbf{C}$ *is a functor* $\mathcal{P} : \mathbf{C}^{op} \to \mathbf{InfSL}$ *where* $\mathbf{InfSL}$ *is the category of finite meet semilattices.*

A primary doctrine is existential *if:*

- $\mathbf{C}$ *has finite products;*
- *the image* $\mathcal{P}_{\pi_C}$ *of any projection* $\pi_C : C \times D \to C$ *admits a left adjoint* $\exists_{\pi_C}$*;*
- *for each pullback like aside, the* Beck-Chevalley condition $\exists_{\pi_{C'}} \circ \mathcal{P}_{1_D \times f} = \mathcal{P}_f \circ \exists_{\pi_C}$ *holds;*

$$\begin{array}{ccc} D \times C' & \xrightarrow{\pi_{C'}} & C' \\ {\scriptstyle 1_D \times f}\downarrow & & \downarrow{\scriptstyle f} \\ D \times C & \xrightarrow[\pi_C]{} & C \end{array}$$

- *for any* $\alpha \in \mathcal{P}(C)$ *and* $\beta \in \mathcal{P}(D \times C)$ *the* Frobenius reciprocity $\exists_{\pi_C}(\mathcal{P}_{\pi_C}(\alpha) \wedge \beta) = \alpha \wedge \exists_{\pi_C}(\beta)$ *holds.*

A hyperdoctrine *is an existential doctrine* $\mathcal{P}$ *such that:*

- $\mathcal{P}$ *factors through the category* $\mathbf{HA}$ *of Heyting algebras and Heyting algebras morphisms;*
- *for all projections* $\pi_C : D \times C \to C$, $\mathcal{P}_{\pi_C}$ *has a right adjoint* $\forall_{\pi_C} : \mathcal{P}(D \times C) \to \mathcal{P}(C)$ *satisfying the Beck-Chevalley condition:* $\forall_{\pi_{C'}} \circ \mathcal{P}_{1_D \times f} = \mathcal{P}_f \circ \forall_{\pi_C}$ *for any* $f : C' \to C$*.*

A primary doctrine, an existential doctrine or a hyperdoctrine, is elementary *if*

- $\mathbf{C}$ *has finite products;*
- *for each object* C *there exists a* fibered equality $\delta_C \in \mathcal{P}(C \times C)$ *such that*

$$\mathcal{P}_{(\pi_1,\pi_2)}(-) \wedge \mathcal{P}_{(\pi_2,\pi_3)}(\delta_C) \dashv \mathcal{P}_{1_D \times \Delta_C}$$

where π_1, π_2 *and* π_3 *are projections* $D \times C \times C \to D \times C$*. This left adjoint will be denoted by* $\exists_{1_D \times \Delta_C}$

▶ Remark 2.2. Usually $\mathbf{C}$ is required to having finite products even in the case of a primary doctrine (cfr. [37]), we will not ask it in order to get the coalgebraic examples in Section 3.

▶ Remark 2.3. Since $\mathbf{C}$ has a terminal object it follows that $\mathcal{P}_{\pi_1}(-) \wedge \delta_C \dashv \mathcal{P}_{\Delta_C}$. This left adjoint will be denoted by $\exists_{\Delta_C}$.

▶ Remark 2.4. In this paper, we work with hyperdoctrines over $\mathbf{HA}$, the category of Heyting algebras and their morphisms; hence the resulting logic is inherently intuitionistic. Clearly, all the development still holds if we restrict ourselves to the subcategory of Boolean algebras $\mathbf{BA}$, yielding a classical version of the logic.

▶ **Example 2.5.** Let $\mathbf{C}$ be a category with finite limits and $(\mathcal{E}, \mathcal{M})$ a stable and proper factorization system on it (see [27]). For every object $C \in \mathbf{C}$ we can define a relation on arrows in $\mathcal{M}$ with codomain C putting $m \leq n$ if and only if there exists t such that $n \circ t = m$. If we ignore size issues this gives us a preorder, from which we get a partial order $\mathcal{M}\text{-}\mathcal{S}ub_{\mathbf{C}}(C)$ by quotienting by the relation $m \simeq n$ if and only if $m \leq n$ and $n \leq m$. The top element is $[1_C]$, while meets are given by pullbacks, and we can pullback any m along any arrow $f : D \to C$ getting an arrow f^*m in $\mathcal{M}$ with codomain D. Summarizing we have a functor $\mathbf{C}^{op} \to \mathbf{InfSL}$ sending C to $\mathcal{M}\text{-}\mathcal{S}ub_{\mathbf{C}}(C)$. This is actually an elementary existential doctrine in which δ_C is the class of the diagonal $C \to C \times C$ (which can be shown to be an element of $\mathcal{M}$) and $\exists_{\pi_C}([m])$ is the $\mathcal{M}$-component of $\pi_C \circ m$, in the sense that it is the class of $n \in \mathcal{M}$ such that $n \circ e = \pi_C \circ m$ for some $e \in \mathcal{E}$ (see [22] for the correspondence between factorization systems and elementary existential doctrines). In general this functor is very far from having Heyting algebras as values but this is the case when $\mathbf{C}$ is a topos and $\mathcal{M}$ the class of all monomorphisms; in this case we get an elementary hyperdoctrine [32].

If we want $\mathcal{M}$ to be the class of all monos we have the following theorem:

▶ **Theorem 2.6** ([23, Th. 4.4.4]). *If* $\mathbf{C}$ *has finite limits then* $\mathcal{Sub}_{\mathbf{C}}$ *is an elementary existential doctrine if and only if* $\mathbf{C}$ *is regular.*

▶ **Proposition 2.7.** *Let* $\mathcal{P} : \mathbf{C}^{op} \to \mathbf{InfSL}$ *be an existential doctrine,* $\mathbf{D}$ *a category with finite products and* $\mathcal{F} : \mathbf{D} \to \mathbf{C}$ *a product preserving functor. Then,* $\mathcal{P} \circ \mathcal{F}^{op}$ *is a existential doctrine. If* $\mathcal{P}$ *is elementary (resp., a hyperdoctrine) then* $\mathcal{P} \circ \mathcal{F}^{op}$ *is elementary (resp., a hyperdoctrine).*

Proof. See proof on page 18. ◀

▶ **Proposition 2.8.** *Let* $\mathcal{P} : \mathbf{C}^{op} \to \mathbf{HA}$ *be an elementary existential doctrine. For every arrow* $f : C \to D$*, the functor* $\mathcal{P}_f$ *has a left adjoint* $\exists_f$ *that satisfies the* Frobenius reciprocity*:* $\exists_f(\mathcal{P}_f(\beta) \wedge \alpha) = \beta \wedge \exists_f(\alpha)$*. If* $\mathcal{P}$ *is a hyperdoctrine then* $\mathcal{P}_f$ *has a right adjoint* $\forall_f$ *too.*

Proof. See proof on page 19. ◀

▶ Remark 2.9. In general these adjoints do not satisfy any form of Beck-Chevalley condition [12, 23, 33, 40].

▶ **Definition 2.10.** *Let* $\mathcal{P} : \mathbf{C}^{op} \to \mathbf{InfSL}$*,* $\mathcal{S} : \mathbf{D}^{op} \to \mathbf{InfSL}$ *be primary doctrines.*

A morphism $\mathcal{P} \to \mathcal{S}$ *is a pair* $(\mathcal{F}, \eta)$ *where* $\mathcal{F} : \mathbf{C} \to \mathbf{D}$ *is a functor and* $\eta : \mathcal{P} \to \mathcal{S} \circ \mathcal{F}^{op}$ *is a natural transformation.*

$(\mathcal{F}, \eta)$ *is a* morphism of elementary doctrines*, or* elementary*, if* $\mathcal{F}$ *preserves finite products and for any object* C *of* $\mathbf{C}$*, it is* $\eta_{C\times C}(\delta_C) = \mathcal{S}_{(\mathcal{F}(\pi_1),\mathcal{F}(\pi_2))}(\delta_{\mathcal{F}(C)})$*.*

$(\mathcal{F}, \eta)$ *is a* morphism of existential doctrine *if* $\mathcal{F}$ *preserves finite products and for any pair of objects* C, D *of* $\mathbf{C}$ *the diagram* (a) *below commutes.*

$$\begin{array}{ccc} \mathcal{P}(D\times C) & \xrightarrow{\exists_{\pi_C}} & \mathcal{P}(C) \\ {\scriptstyle \eta_{D\times C}}\downarrow & & \\ \mathcal{S}(\mathcal{F}(D\times C)) & & \Big\downarrow{\scriptstyle \eta_C} \\ {\scriptstyle \mathcal{S}_{(\mathcal{F}(\pi_D),\mathcal{F}(\pi_C))}}\downarrow & & \\ \mathcal{S}(\mathcal{F}(D)\times\mathcal{F}(C)) & \xrightarrow{\exists_{\pi_{\mathcal{F}(C)}}} & \mathcal{S}(\mathcal{F}(D)) \end{array} \qquad \begin{array}{ccc} \mathcal{P}(D\times C) & \xrightarrow{\forall_{\pi_C}} & \mathcal{P}(C) \\ {\scriptstyle \eta_{D\times C}}\downarrow & & \\ \mathcal{S}(\mathcal{F}(D\times C)) & & \Big\downarrow{\scriptstyle \eta_C} \\ {\scriptstyle \mathcal{S}_{(\mathcal{F}(\pi_D),\mathcal{F}(\pi_C))}}\downarrow & & \\ \mathcal{S}(\mathcal{F}(D)\times\mathcal{F}(C)) & \xrightarrow{\forall_{\pi_{\mathcal{F}(C)}}} & \mathcal{S}(\mathcal{F}(D)) \end{array}$$

(a) *(b)*

$(\mathcal{F}, \eta)$ *is a* morphism of hyperdoctrines *if it is a morphism of existential doctrine, the diagram* (b) *above commutes too and each component of* η *preserves finite suprema and implication.*

If (F, η) *is also elementary then we call it a* morphism of elementary existential doctrines *or of* elementary hyperdoctrines*.*

Let $(\mathcal{F}, \eta), (\mathcal{G}, \epsilon) : \mathcal{P} \to \mathcal{S}$ *be two morphisms; a* 2*-arrow* $(\mathcal{F}, \eta) \to (\mathcal{G}, \epsilon)$ *is a natural transformations* $\theta : \mathcal{F} \to \mathcal{G}$ *such that* $\eta_C(\alpha) \leq \mathcal{S}_{\theta_C}(\epsilon_C(\alpha))$*.*

This defines the 2*-categories* $\mathbf{PD}$*,* $\mathbf{ED}$*,* $\mathbf{HD}$ *of primary doctrines, existential doctrines and hyperdoctrines, and the subcategories* $\mathbf{EPD}$*,* $\mathbf{EED}$*,* $\mathbf{EHD}$ *of their elementary variants.*

2.2 Closure operators on doctrines

In this section we introduce the key notion of closure operators on doctrines.

▶ **Definition 2.11.** *Let $\mathcal{P}$ be a doctrine. A* closure operator *on $\mathcal{P}$ is a (possibly large) family $\mathfrak{c} = \{\mathfrak{c}_C\}_{C \in \mathbf{Ob(C)}}$ of functions $\mathfrak{c}_C : \mathcal{P}(C) \to \mathcal{P}(C)$ such that:*

- *for any object C, $\mathfrak{c}_C$ is monotone and* inflationary*, i.e., $1_{\mathcal{P}(C)} \leq \mathfrak{c}_C$*
- *any arrow $f : C \to D$ is* continuous*, i.e. $\mathfrak{c}_C \circ \mathcal{P}_f \leq \mathcal{P}_f \circ \mathfrak{c}_D$.*

A closure operator $\mathfrak{c}$ is said to be

- grounded *if $\mathfrak{c}_C(\bot) = \bot$ for all objects C such that $\mathcal{P}(C)$ has a minimum;*
- additive *if $\mathfrak{c}_C(\alpha \vee \beta) = \mathfrak{c}_C(\alpha) \vee \mathfrak{c}_C(\beta)$ for all objects C such that $\mathcal{P}(C)$ has binary suprema;*
- finitely additive *if it is grounded and additive;*
- full additive *if $\mathfrak{c}_C(\bigvee_{i \in I} \alpha_i) = \bigvee_{i \in I} \mathfrak{c}_C(\alpha_i)$ for all $I \neq \emptyset$ and C such that $\mathcal{P}(C)$ has I-indexed suprema;*
- idempotent *if $\mathfrak{c}_C \circ \mathfrak{c}_C = \mathfrak{c}_C$ for all object C.*

A closure doctrine *is a pair $(\mathcal{P}, \mathfrak{c})$ where $\mathcal{P}$ is a primary doctrine and $\mathfrak{c}$ a closure operator on it. We say that $(\mathcal{P}, \mathfrak{c})$ is* elementary, existential*, or a* hyperdoctrine*, if $\mathcal{P}$ is.*

▶ Remark 2.12. Continuity can be interpreted as a form of oplax naturality [20], even if $\mathfrak{c}_C$ is not an arrow of **InfSL** in general.

▶ **Example 2.13.** Lawvere-Tierney topologies on a topos provide examples of idempotent closure operators on the elementary hyperdoctrine of subobjects [6, 26, 32].

▶ Remark 2.14. Full additivity does not imply groundedness since we only ask for preservation of suprema indexed on non empty sets.

▶ **Proposition 2.15.** *Let $\mathcal{P}$ be a doctrine and $f : C \to D$ a morphism such that $\mathcal{P}_f$ has a left adjoint $\exists_f$, then for every closure operator $\mathfrak{c}$ on $\mathcal{P}$ continuity of f is equivalent to*

$$\exists_f \circ \mathfrak{c}_C \leq \mathfrak{c}_D \circ \exists_f$$

Proof. Let's compute:

$$\mathfrak{c}_C \circ \mathcal{P}_f \leq \mathcal{P}_f \circ \exists_f \circ \mathfrak{c}_C \circ \mathcal{P}_f \leq \mathcal{P}_f \circ \mathfrak{c}_D \circ \exists_f \circ \mathcal{P}_f \leq \mathcal{P}_f \circ \mathfrak{c}_D$$
$$\exists_f \circ \mathfrak{c}_C \leq \exists_f \circ \mathfrak{c}_C \circ \mathcal{P}_f \circ \exists_f \leq \exists_f \circ \mathcal{P}_f \circ \mathfrak{c}_D \circ \exists_f \leq \mathfrak{c}_D \circ \exists_f$$

◀

If we think of a morphism of (primary, existential, elementary, hyper)doctrines $(\mathcal{F}, \eta) : \mathcal{P} \to \mathcal{Q}$ as a "translation" of "types" and "predicates" then, when closure operators are available, it is natural to ask for this "translation" to take place in a continuous way.

▶ **Definition 2.16.** *A* morphism of closure (elementary, existential, hyper)doctrines *$(\mathcal{F}, \eta) : (\mathcal{P}, \mathfrak{c}) \to (\mathcal{Q}, \mathfrak{d})$ is a morphism of (elementary, existential, hyper)doctrines $\mathcal{F} : \mathcal{P} \to \mathcal{Q}$ such that η is* continuous*, i.e. $\mathfrak{d}_{\mathcal{F}(C)} \circ \eta_C \leq \eta_C \circ \mathfrak{c}_C$ for all C. We say that $(\mathcal{F}, \eta)$ is* open *if equality holds for all the objects C. A 2-cell $\theta : (\mathcal{F}, \eta) \to (\mathcal{G}, \epsilon)$ is defined as in the case of doctrines. In this way we get the 2-categories* **cPD**, **cED**, **cHD** *of closure doctrines, closure existential doctrines, closure hyperdoctrines and the subcategories* **cEPD**, **cEED**, **cEHD** *of their elementary variants.*

3 Examples of closure hyperdoctrines

3.1 Topological examples

As a first class of examples, we introduce three closure hyperdoctrines starting from the usual category **Top** of topological spaces and continuous maps. The first one corresponds to the *closure spaces* used in, e.g., [10, 11, 18].

▶ **Definition 3.1.** *The category* **PrTop** *of* pretopological spaces *(or* closure spaces*) is the category in which:*

- *objects are pairs* $(X, \mathfrak{c})$ *of a set* X *and a monotone function* $\mathfrak{c} : \mathcal{P}(X) \to \mathcal{P}(X)$ *such that* $1_{\mathcal{P}(X)} \leq \mathfrak{c}$ *and* $\mathfrak{c}$ *preserves finite (even empty) suprema;*
- *an arrow* $f : (X, \mathfrak{c}_X) \to (Y, \mathfrak{c}_Y)$ *is a function* $f : X \to Y$ *such that* $f^{-1} : (\mathcal{P}(Y), \mathfrak{c}_Y) \to (\mathcal{P}(X), \mathfrak{c}_X)$ *is continuous.*

Another example is given by so called *convergence spaces* (cfr. [14]).

▶ **Definition 3.2.** *For any set* X *let* $\mathbf{Fil}(X)$ *be the set of proper filters (i.e.,* $\emptyset$ *is not among them) on it. The category* **FC** *of* filter convergence spaces *is the category in which:*

- *an object is a pair* (X, q_X) *given by a set* X *and a function* $q_X : X \to \mathcal{P}(\mathbf{Fil}(X))$ *such that, for any* $x \in X$, $q_X(x)$ *is upward closed and* $\dot{x} := \{A \subset X \mid x \in A\}$ *belongs to* $q_X(x)$.
- *an arrow* $f : (X, q_x) \to (Y, q_Y)$ *is a function* $f : X \to Y$ *such that the filter* $f(F)$ *generated by the images of* F*'s elements belongs to* $q_Y(f(x))$ *whenever* $F \in q_X(x)$.

▶ **Proposition 3.3.** *The obvious forgetful functors from* **Top**, **PrTop** *and* **FC** *to* **Set** *preserve finite products.*

Proof. For **Top** it is clear, for the other two categories see [14, Ch.3]. ◀

By Proposition 2.7 and the previous one, we have three elementary hyperdoctrines

$$\mathcal{P}^t : \mathbf{Top}^{op} \to \mathbf{HA} \quad \mathcal{P}^p : \mathbf{PrTop}^{op} \to \mathbf{HA} \quad \mathcal{P}^f : \mathbf{FC}^{op} \to \mathbf{HA}$$

which we now endow with closure operators.

▶ **Definition 3.4.** *We define the following closure operators:*

1. *the* Kuratowski closure operator $k = \{k_{(X,\theta)}\}_{(X,\theta)\in\mathbf{Ob}(\mathbf{Top})}$ *on* $\mathcal{P}^t$ *where* $k_{(X,\theta)}$ *is the closure operator associated with the topology* θ*;*
2. *the* Čech closure operator $c = \{c_{(X,\mathfrak{c})}\}_{(X,\mathfrak{c})\in\mathbf{Ob}(\mathbf{PrTop})}$ *on* $\mathcal{P}^p$ *where* $c_{(X,\mathfrak{c})}$ *is just* $\mathfrak{c}$*;*
3. *the* Katětov closure operator $\mathfrak{k} = \{\mathfrak{k}_{(X,q_X)}\}_{(X,\mathfrak{q}_\mathfrak{X})\in\mathbf{Ob}(\mathbf{FC})}$ *on* $\mathcal{P}^f$ *where*

$$\mathfrak{k}_{(X,q_X)} : \mathcal{P}(X) \to \mathcal{P}(X) \qquad A \mapsto \{x \in X \mid \exists F \in q_X(x).A \in F\}$$

▶ **Proposition 3.5** ([14, Ch. 3])**.**

1. *k, c and $\mathfrak{k}$ are grounded and additive closure operators, moreover k is idempotent.*
2. *There exists a sequence of inclusion functors* $\mathbf{Top} \xrightarrow{i} \mathbf{PrTop} \xrightarrow{j} \mathbf{FC}$ *each of which has a left adjoint.*
3. *We have a sequence* $(\mathcal{P}^t, k) \xrightarrow{(i,\eta)} (\mathcal{P}^p, c) \xrightarrow{(j,\epsilon)} (\mathcal{P}^f, \mathfrak{k})$ *of morphisms in* **cEHD** *where* η *and* ϵ *have identities as components.*

Proof.

1. For k and c the proposition is obvious, let us examine $\mathfrak{k}$: since $\dot{x} \in q_X(x)$ then $A \subset \mathfrak{k}_X(A)$, if $A \subset B$ then any filters that contains the former contains the latter too and this implies monotonicity, groundedness follows from the fact that $\emptyset$ does not belong to any proper filter, for additivity we can complete any filter $\mathcal{F}$ to which $A \cup B$ belong to an ultrafilter $\mathcal{U}$ that belongs to $q_X(x)$ since the latter is upward closed, either A or B must belong to $\mathcal{U}$ and we are done.
2. i sends a topological space to the pretopological space given by the closure operator associate to its topology, j sends $(X, \mathfrak{c})$ to $(X, q^{\mathfrak{c}}_X)$ where

$$q^{\mathfrak{c}}_X : X \to \mathcal{P}(\mathbf{Fil}(X)) \qquad x \mapsto \{\mathcal{F} \in \mathbf{Fil}(X) \mid \mathcal{V}_x \subset \mathcal{F}\}$$

where $\mathcal{V}_x := \{S \subset X \mid x \notin \mathfrak{c}(X \setminus S)\}$. For the left adjoints see [14].
3. This is obvious. ◀

For other examples of closure operators on topological spaces we refer the reader to [14].

3.2 Algebraic examples

▶ **Proposition 3.6.** *Let* **Grp** *be the category of groups and* **CRing** *that of commutative, unital rings (where we require that $f(1_A) = 1_B$ for any $f : A \to B$). Then, $\mathcal{Sub}_{\mathbf{Grp}}$ and $\mathcal{Sub}_{\mathbf{CRing}}$ are elementary existential doctrines.*

Proof. This follows at once from Theorem 2.6. ◀

▶ Remark 3.7. Notice that, even if $\mathcal{Sub}_{\mathbf{Grp}}(G)$ and $\mathcal{Sub}_{\mathbf{CRing}}(A)$ admit finite suprema for any group G or commutative ring A with unity, preimages do not preserve them in general: for instance they do not preserve the bottom subobject. Then $\mathcal{Sub}_{\mathbf{Grp}}$ or $\mathcal{Sub}_{\mathbf{CRing}}$ cannot be universal doctrines.

The following examples are taken from [14].

▶ **Definition 3.8** (Groups)**.** *The* normal closure *on a group G is given by*

$$\nu_G : \mathcal{Sub}_{\mathbf{Grp}}(G) \to \mathcal{Sub}_{\mathbf{Grp}}(G) \qquad H \mapsto \bigcap \{N \leq G \mid H \leq N \trianglelefteq G\}$$

where we have chosen the image of a monomorphism as a canonical representative of it.

▶ **Proposition 3.9.** *The family previous defined forms a closure operators ν on $\mathcal{Sub}_{\mathbf{Grp}}$ that is idempotent, fully additive and grounded.*

Proof. Since the preimage of a normal subgroup is normal we have that the ν actually exists as a closure operator. The three poperties of it follow immediately by the fact that $\{0\}$ is normal and so are the arbitrary intersections or sums of normal subgroups. ◀

▶ **Definition 3.10** (Rings)**.** *Let A be a unital commutative ring and B a subring, we define* $\mathsf{int}_A(B)$ *to be the* integral closure of B*:*

$$\mathsf{int}_A(B) := \{a \in A \mid p(a) = 0 \text{ for some } p \in B[x]\}$$

Again we are denoting a subobject by the image of any representative of it.

▶ **Proposition 3.11.** *For any A int_A is a function $\mathcal{Sub}_{\mathbf{CRing}}(A) \to \mathcal{Sub}_{\mathbf{CRing}}(A)$, moreover the family of this functions forms an idempotent closure operator* int*.*

Proof. To show that $\mathsf{int}_A(B)$ is a subring of A and idempotency we refer to [2, Cor. 5.3, 5.5]. Let us show that int is actually a closure operator. Consider $f : A \to B$ and C a subring of B, let $a \in A$ such that $p(a) = 0$ for some $p \in f^{-1}(C)[X]$ with coefficients $\{p_i\}_{i=0}^{\deg(p)}$, then $q(f(a)) = 0$ where $q \in C[X]$ has coefficients $\{f(p_i)\}_{i=0}^{\deg(p)}$ and we are done. ◀

3.3 Contact algebras

▶ **Definition 3.12** ([15, 16]). *A* contact algebra *is an Heyting algebra H equipped with a symmetric binary relation C such that*

- *if xCy then x and y are different from $\bot$;*
- *if $x \neq \bot$ then xCx;*
- *if $xC(y \vee z)$ if and only if xCy or xCz.*

(H, C) is complete *if H is so. A morphism $f : (H, C) \to (K, D)$ is a morphism of Heyting algebras $f : H \to K$ such that $(f \times f)(C) \subset D$.* **CA** *denotes the category of contact algebras and* **CCA** *its subcategory of complete contact algebras and morphisms preserving all suprema.*

For a complete contact algebra (H, C) and $x \in H$, we define C_x to be the set $\{y \in H \mid xCy\}$ and the contact closure *on (H, C) as*

$$\mathfrak{c}_{(H,C)} : H \to H \qquad x \mapsto x \vee \sup(C_x)$$

▶ Remark 3.13. Clearly the third condition of the definition of contact algebra can be rephrased as $C_{x\vee y} = C_x \cup C_y$.

▶ Remark 3.14. Let $f : (H, C) \to (K, D)$ be an arrow of **CCA**, by the adjoint functor theorem ([7]) it has a right adjoint f^*. If we regard H and K as meet-semilattices then $f^* : K \to K$, being a right adjoint, is an arrow of **InfSL**.

▶ **Proposition 3.15.** *$\mathfrak{c}$ is a closure operator on the doctrine $\mathcal{U} : \mathbf{CCA}^{op} \to \mathbf{InfSL}$ sending*

$$\begin{array}{ccc} (H, C) & \longmapsto & H \\ f \downarrow & & \uparrow f^* \\ (K, D) & \longmapsto & K \end{array}$$

Proof. Let (H, C) be a contact algebra and $x, y \in H$. Clearly $x \leq \mathfrak{c}_{(H,C)}(x)$, if $x \leq y$ then $y = x \vee y$ and, by the previous remark $C_x \leq C_y$ so that $\mathfrak{c}_{(H,C)}(x) \leq \mathfrak{c}_{(H,C)}(y)$. Let $f : (H, C) \to (K, D)$ be an arrow of **CCA**, then, for every $x \in H$, $f(\sup(C_x)) = \sup(f(C_x))$. Now, if $y \in f(C_x)$ then $y = f(z)$ for some z such that zCx, so $yDf(x)$ and thus $f(C_x) \subset D_{f(x)}$, we can conclude that $f(\sup(C_x)) \leq \sup(D_{f(x)})$ from which

$$f(\mathfrak{c}_{(H,C)}(x)) \leq \mathfrak{c}_{(K,D)}(f(x))$$

and we can conclude by Proposition 2.15. ◀

3.4 A representable example

▶ **Theorem 3.16.** *For any complete Heyting algebra H, the functor $\mathbf{Set}(-, H) : \mathbf{Set}^{op} \to \mathbf{HA}$ is an elementary hyperdoctrine.*

Proof. See, for instance, [39, Section 2.2]. ◀

▶ **Corollary 3.17.** *$\mathbf{Set}(-, [0, 1]) : \mathbf{Set}^{op} \to \mathbf{HA}$ is an elementary hyperdoctrine on $\mathbf{Set}$.*

▶ **Definition 3.18.** *For any fixed $\epsilon \in [0, 1]$, and any set X, we define, for an $f : X \to [0, 1]$:*

$$\mathfrak{c}_{X,\epsilon}(f) : X \to [0, 1] \qquad x \mapsto f(x) \dot{+} \epsilon$$

where

$$\dot{+} : [0, 1] \times [0, 1] \to [0, 1] \qquad (t, s) \mapsto \max(t + s, 1)$$

In this way we get a function

$$\mathfrak{c}_{X,\epsilon} : \mathbf{Set}(X, [0, 1]) \to \mathbf{Set}(X, [0, 1]) \qquad f \mapsto \mathfrak{c}_{X,\epsilon}(f)$$

▶ **Proposition 3.19.** *For any $\epsilon \geq 0$, the collection $\mathfrak{c}_\epsilon$ of all the functions $\mathfrak{c}_{X,\epsilon}$ is a closure operator.*

Proof. Clearly $f \leq \mathfrak{c}_{X,\epsilon}(f)$ for any $f : X \to [0,1]$, monotonicity is clear, let's check continuity of any function $g : X \to Y$:

$$\mathfrak{c}_{X,\epsilon}(f \circ g)(x) = (f \circ g)(x) \dot{+} \epsilon = f(g(x)) \dot{+} \epsilon = \mathfrak{c}_{x,\epsilon}(f)(g(x)) = (\mathfrak{c}_{x,\epsilon}(f) \circ g)(x) \qquad \blacktriangleleft$$

▶ Remark 3.20. $\mathfrak{c}_\epsilon$ is not grounded if $\epsilon \neq 0$ (in that case it reduces to the discrete closure operator) but it is additive.

3.5 Fuzzy sets

We can refine the previous example considering *fuzzy sets*.

▶ **Definition 3.21.** *[41, 42] The category* **Fzs** of fuzzy sets *has:*
- *pairs (A, α) with $\alpha : A \to [0,1]$ as objects;*
- *as arrows $f : (A, \alpha) \to (B, \beta)$ functions $f : A \to B$ such that $\alpha(x) \leq \beta(f(x))$.*

▶ **Definition 3.22.** *A* fuzzy subset *of (A, α) is a function $\xi : A \to [0,1]$ such that $\xi(x) \leq \alpha(x)$ for all $x \in A$.*
Let us summarize some results about **Fzs**.

▶ **Proposition 3.23.** 1. **Fzs** *is a quasitopos;*
2. *there exists a proper and stable factorization system given by strong monomorphisms and epimorphisms;*
3. *fuzzy subsets of (A, α) correspond to equivalence of strong monomorphisms of codomain (A, α);*
4. *the functor $\mathbf{Fzs}^{op} \to \mathbf{HA}$ assigning to each (A, α) the set of its fuzzy subsets and $f : (A, \alpha) \to (B, \beta)$ to the function f^* defined by:*

$$f^*(\xi) : A \to [0,1] \qquad x \mapsto \alpha(x) \wedge \xi(f(x))$$

is an elementary hyperdoctrine.

Proof. See [41, Ch. 8]. Explicitly the hyperdoctrine structure is given by:

$$\exists_f(\xi) : B \to [0,1] \qquad \forall_f(\xi) : B \to [0,1]$$
$$y \mapsto \bigvee_{x \in f^{-1}(y)} \xi(x) \qquad y \mapsto \beta(y) \wedge \bigwedge_{x \in f^{-1}(y)} (\alpha(x) \Rightarrow \xi(x))$$

for any $f : (A, \alpha) \to (B, \beta)$ and $\xi \in \mathcal{FzSub}(A, \alpha)$. $\blacktriangleleft$

▶ **Proposition 3.24.** *Let $\mathcal{E} = \{\epsilon_{(A,\alpha)}\}_{(A,\alpha) \in \mathbf{Ob(Fzs)}}$ be a family of functions $\epsilon_{(A,\alpha)} : (A, \alpha) \to [0,1]$ such that $\epsilon_{(A,\alpha)}(x) \leq \epsilon_{(B,\beta)}(f(x))$ for any $f : (A, \alpha) \to (B, \beta)$. Then*

$$\mathfrak{c}^{\mathcal{E}}_{(A,\alpha)} : \mathcal{FzSub}(A, \alpha) \to \mathcal{FzSub}(A, \alpha) \qquad \xi \mapsto (\xi + \epsilon_{(A,\alpha)}) \wedge \alpha$$

gives us an additive closure operator on $\mathcal{FzSub}$.

Proof. See proof on page 20. $\blacktriangleleft$

▶ Remark 3.25. $\mathfrak{c}^{\mathcal{E}}$ is not grounded in general.

The condition on the elements of $\mathcal{E}$ is very restrictive. In fact, it can be eased restricting to a suitable subclass of arrows and using the following lemma.

▶ **Lemma 3.26.** *Let $\mathcal{P} : \mathbf{C}^{op} \to \mathbf{InfSL}$ be a doctrine, and $\mathfrak{c} = \{\mathfrak{c}_C : \mathcal{P}(C) \to \mathcal{P}(C)\}_{C \in \mathbf{Ob(C)}}$ be a family of monotone and inflationary operators. Let $\mathcal{A}$ be a (possibly large) family of $\mathbf{C}$-arrows such that:*

- *$\mathcal{A}$ is closed under composition;*
- *if $f \in \mathcal{A}$ then $1_{\mathrm{dom}(A)}$ and $1_{\mathrm{cod}(A)}$ are in $\mathcal{A}$;*
- *$f : C \to D$ in $\mathcal{A}$ implies $\mathfrak{c}_C \circ \mathcal{P}_f \leq \mathcal{P}_f \circ \mathfrak{c}_D$.*

Then $\mathcal{P}$ induces a doctrine $\mathcal{P}^{\mathcal{A}}$ on the subcategory $\mathbf{C}_{\mathcal{A}}$ induced by $\mathcal{A}$ for which $\mathfrak{c} = \{\mathfrak{c}_C\}_{C \in \mathbf{Ob(C}_{\mathcal{A}})}$ is a closure operator. Moreover, if for all f, g in $\mathcal{A}$ also (f, g) and the projections from $\mathrm{cod}(f) \times \mathrm{cod}(g)$ are in $\mathcal{A}$, then $\mathcal{P}^{\mathcal{A}}$ is existential, elementary or an hyperdoctrine if $\mathcal{P}$ is.

Proof. This is almost tautological since the condition on $\mathcal{A}$ guarantee that the inclusion functor $\mathbf{C}_{\mathcal{A}}$ preserves limits and we can use Proposition 2.7. ◀

3.6 Coalgebraic examples

▶ **Definition 3.27** ([24, 29])**.** *Let $\mathbf{C}$ be a category with finite products and $\mathcal{F} : \mathbf{C} \to \mathbf{C}$ an endofunctor. The category $\mathbf{CoAlg}(\mathcal{F})$ of* coalgebras for $\mathcal{F}$ *has*

- *arrows $\gamma_C : C \to \mathcal{F}(C)$ as objects;*
- *arrows $f : C \to D$ such that $\gamma_D \circ f = \mathcal{F}(f) \circ \gamma_C$ as morphisms $f : \gamma_C \to \gamma_D$.*

Notice that in general $\mathbf{CoAlg}(\mathcal{F})$ is not complete and products in it can be very different from products in $\mathbf{C}$ [21], so it does not make much sense to look for an existential doctrine on it. However, for $\mathbf{Set}$-based coalgebras we get a primary doctrine $\mathcal{P}^c : \mathbf{CoAlg}(\mathcal{F})^{op} \to \mathbf{InfSL}$ composing the contravariant power object $\mathcal{P} : \mathbf{Set}^{op} \to \mathbf{InfSL}$ with the opposite of the obvious forgetful functor $\mathbf{CoAlg}(\mathcal{F}) \to \mathbf{Set}$.

▶ **Definition 3.28.** *Let $\mathcal{F} : \mathbf{C} \to \mathbf{C}$ be a functor and $\mathcal{P}$ a primary doctrine on $\mathbf{C}$. A* predicate lifting *is a natural transformation $\Box : \mathcal{U} \circ \mathcal{P} \to \mathcal{U} \circ \mathcal{P} \circ \mathcal{F}^{op}$ where $\mathcal{U}$ is the forgetful functor $\mathbf{InfSL} \to \mathbf{Poset}$.*

▶ Remark 3.29. A similar notion can be found in [25]. In particular, the predicate liftings of Examples 3.32 and 3.34 below fit Jacobs and Sokolova's framework.

▶ **Definition 3.30.** *For any predicate lifting $\Box$, we define two closure operators on $\mathcal{P}^c$.*

1. *For any coalgebra $\gamma_X : X \to \mathcal{F}(X)$, notice that $\mathcal{P}^c(\gamma_X) = \mathcal{P}(X)$; hence we can define*

$$\mathsf{pre}_{\gamma_X} : \mathcal{P}(X) \to \mathcal{P}(X) \qquad \alpha \mapsto \alpha \vee \mathcal{P}_{\gamma_X}(\Box_X(\alpha))$$

2. *Suppose that $\mathcal{P}$ admits arbitrary meets; for $\gamma_X : X \to \mathcal{F}(X)$ and $\alpha \in \mathcal{P}(X)$ we define*

$$\mathcal{N}_{\gamma_X}(\alpha) := \{\beta \in \mathcal{P}(X) \mid \alpha \leq \mathcal{P}_{\gamma_X}(\Box_X(\beta))\} \quad \mathsf{s}_{\gamma_X}(\alpha) := \inf(\mathcal{N}_{\gamma_X}(\alpha))$$

and

$$\mathsf{suc}_{\gamma_X} : \mathcal{P}(X) \to \mathcal{P}(X) \qquad \alpha \mapsto \alpha \vee \mathsf{s}_{\gamma_X}(\alpha)$$

▶ **Lemma 3.31.** *Let $\mathcal{F} : \mathbf{C} \to \mathbf{C}$ be a functor and $\Box$ a predicate lifting, then:*

1. *$\{\mathsf{pre}_{\gamma_X}\}_{\gamma_X \in \mathbf{Ob(CoAlg}(\mathcal{F}))}$ defines a closure operator pre on $\mathcal{P}^c$.*
2. *$\mathsf{s}_{\gamma_X}(\alpha)$ is the minimum of $\mathcal{N}_{\gamma_X}(\alpha)$ whenever $\mathcal{P}$ has arbitrary meets and, for any coalgebra $\gamma_X : X \to \mathcal{F}(X)$, $\mathcal{P}_{\gamma_X}$ and $\Box_X$ commute with them;*
3. *in the hypothesis above if $\mathcal{P}_f$ commutes with arbitrary infima for all arrows f then $\{\mathsf{suc}_{\gamma_X}\}_{\gamma_X \in \mathbf{Ob(CoAlg}(\mathcal{F}))}$ defines a closure operators suc on $\mathcal{P}^c$.*

Proof. See proof on page 21. ◀

The previous result provides us with many examples with practical applications.

▶ **Example 3.32.** Let $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ be the covariant powerset functor, and $\mathcal{P} : \mathbf{Set}^{op} \to \mathbf{InfSL}$ be the controvariant one, seen as primary doctrine. We can define a predicate lifting $\Box$ taking as components:

$$\Box_X : \mathcal{P}(X) \to \mathcal{P}(\mathcal{P}(X)) \qquad A \mapsto \downarrow A$$

where $\downarrow A$ denotes the set of downward-closed subsets of A. In this case for any coalgebra $\gamma_X : X \to \mathcal{P}(X)$ we have

$$x \in \gamma_X^{-1}(\Box_X(A)) \iff \gamma_X(x) \subset A$$
$$B \in \mathcal{N}_{\gamma_X}(A) \iff \gamma_X(a) \subset B \text{ for any } a \in A$$

so $\mathsf{s}_{\gamma_X}(A) = \bigcup_{a \in A} \gamma_X(a)$ and $\mathsf{suc}_{\gamma_X}(A) = A \cup \bigcup_{a \in A} \gamma_X(a)$.

By this description it is clear that suc is grounded and fully additive. pre is grounded too but it is not even finitely additive: take $4 := \{0, 1, 2, 3\}$ with stuctural map γ_4 given by

$$0 \mapsto \{3\} \quad 1 \mapsto \{2, 3\} \quad 2 \mapsto \{2\} \quad 3 \mapsto \{3\}$$

Now take $A := \{2, 3\}$, it is immediate to see that $\mathsf{pre}_{\gamma_4}(A) = 4$, on the other hand $\mathsf{pre}_{\gamma_4}(\{2\}) = \{2\}$ and $\mathsf{pre}_{\gamma_4}(\{3\}) = \{0, 3\}$.

▶ Remark 3.33. In this case, pre and suc meanings (and notation) become clearer: if we think to the value of $\gamma_X(x)$ as the family of points accessible from $x \in X$ then pre_{γ_X} adds to a subset A the set of its *predecessors*, i.e. points from which some $a \in A$ is accessible, while suc_{γ_X} adds the set of *successors*, i.e. points which are accessible from some point of A.

▶ **Example 3.34** (Probabilistic frames [3, 4, 19])**.** Let **Meas** be the category of measurable space and measurable functions; then we can take as primary doctrine $\mathcal{P}$ the functor

$$\begin{array}{ccc} (X, \Omega_X) & \longmapsto & \Omega_X \\ f \downarrow & & \downarrow f^{-1} \\ (Y, \Omega_Y) & \longmapsto & \Omega_Y \end{array}$$

As endofunctor we can take the *Giry monad* $\mathcal{G} : \mathbf{Meas} \to \mathbf{Meas}$:

- given an object (X, Ω_X), $\mathcal{G}(X, \Omega_X)$ is the set of all probability measures on Ω_X equipped with the smallest σ-algebra for which all the *evaluation functions*

  $$\mathsf{ev}_A : \mathcal{G}(X, \Omega_X) \to [0, 1] \qquad \mu \mapsto \mu(A)$$

  with $A \in \Omega_X$, are Borel-measureable.
- for a measurable $f : (X, \Omega_X) \to (Y, \Omega_Y)$ we can define

  $$\mathcal{G}(f) : \mathcal{G}(X, \Omega_X) \to \mathcal{G}(Y, \Omega_Y) \qquad \mu \mapsto \mu \circ f^{-1}$$

Given a coalgebra $\gamma_{(X,\Omega_X)}$ and $p \in [0, 1]$ we can now define

$$\Box_{(X,\Omega_X),p} : \Omega_X \to \mathcal{P}(\mathcal{G}(X)) \qquad A \mapsto \{\mu \in \mathcal{G}(X, \Omega_X) \mid \mu(A) \geq p\}$$

Notice that the set on the right is $\mathsf{ev}_A^{-1}([p, 1])$ and so $\Box_{(X,\Omega_X),p}$ is well defined. In this situation we have

$$\mathsf{pre}_{\gamma_{(X,\Omega_X)}}(A) := A \cup \{x \in X \mid p \leq \gamma_{(X,\Omega_X)}(x)(A)\}$$

▶ Remark 3.35. If we think of a coalgebra $\gamma_{(X,\Omega_X)}$ as describing how likely is a transition from a state to the various $A \in \Omega_X$ then, given a $p \in [0, 1]$, $\mathsf{pre}_{\gamma_{(X,\Omega_X)}}(A)$ is the set of points which access A with probability at least p.

4 Logics for Closure Operators

In this section, we provide a sound and complete logic for closure hyperdoctrines. This logic is a (first order) version of Spatial Logic for Closure Spaces (SLCS) [11], although with a slightly different presentation.

4.1 Syntax and derivation rules

We briefly recall the categorical presentation of signatures, as in [23].

▶ **Definition 4.1.** *A* signature Σ *is a triple* $(|\Sigma|, \Gamma, \Pi)$ *where*

- $|\Sigma|$ *is a set, called the set of* basic types*;*
- Γ *is a functor*[2] $|\Sigma|^\star \times |\Sigma| \to \mathbf{Sets}$*. We will call* function symbol *an element* f *of* $\Gamma((\sigma_1, \dots, \sigma_n), \sigma_{n+1})$ *and we will write* $f : \sigma, \dots, \sigma_n \to, \sigma_{n+1}$*;*
- Π *is a functor* $|\Sigma|^\star \to \mathbf{Set}$*, we will call* predicate symbol *an element* P *of* $\Pi(\sigma_1, \dots, \sigma_n)$ *and we will write* $P : \sigma_1, \dots, \sigma_n$*.*

A morphism of signatures $\phi : \Sigma_1 \to \Sigma_2$ *is a triple* (ϕ_1, ϕ_2, ϕ_3) *such that*

- ϕ_1 *is a function* $|\Sigma_1| \to |\Sigma_2|$*;*
- ϕ_2 *is a natural transformation* $\Gamma_1 \to \Gamma_2 \circ (\phi_1^\star \times \phi_1)$*;*
- ϕ_3 *is a natural transformation* $\Pi_1 \to \Pi_2 \circ \phi_1^\star$*.*

For any $\sigma \in |\Sigma|$ *we fix an countably infinite set* X_σ *of* variables*; definition of terms is straightforward ([23]).*

▶ **Definition 4.2.** *Given a signature* Σ*, its* classifying category $\mathbf{Cl}(\Sigma)$ *is such that*

- *objects are contexts;*
- *Given* $\Gamma := [x_i : \sigma_i]_{i=1}^n$ *and* $\Delta = [y_i : \tau_i]_{i=1}^m$ *an arrow* $\Gamma \to \Delta$ *is a* m*-uple of terms* $(T_1, ..., T_m)$ *such that* $\Gamma \vdash T_i : \tau_i$ *for any* i*;*
- *composition is given by substitution.*

▶ **Proposition 4.3.** $\mathbf{Cl}(\Sigma)$ *is a category with finite products for any signature* Σ*.*

Proof. Associativity of composition and the fact that $(x_1, ..., x_n)$ is the identity for $[x_i : \sigma_i]_{i=1}^n$ follows from a straightforward computation. The empty context is clearly terminal while, given two contexts $\Gamma := [x_i : \sigma_i]_{i=1}^n$ and $\Delta = [y_i : \tau_i]_{i=1}^m$ we can take their concatenation as a product $\Gamma \times \Delta$, the universal property follows immediately. ◀

Now we can introduce the rules for context and closure operators of the Spatial Logic for Closure Spaces, over any given signature.

As usual, we denote by $\Gamma \vdash t : \tau$ the judgment "t has type τ in context Γ", and by $\Gamma \vdash \phi : \mathsf{Prop}$ the judgment "ϕ is a well-formed formula in context Γ".

▶ **Definition 4.4.** *The rules for contexts and well-formed formulae for the closure operators for a signature* Σ *are the usual ones for a first order signature (see [23]) plus:*

$$\frac{\Gamma \vdash \phi : \mathsf{Prop}}{\Gamma \vdash \mathcal{C}(\phi) : \mathsf{Prop}}\ \mathcal{C}\text{-}F \qquad \frac{\Gamma \vdash \phi : \mathsf{Prop} \qquad \Gamma \vdash \psi : \mathsf{Prop}}{\Gamma \vdash \phi \mathcal{U} \psi : \mathsf{Prop}}\ \mathcal{U}\text{-}F$$

For any context Γ *we define* $\mathbf{Form}_\Sigma(\Gamma)$ *to be the set of formulae* ϕ *such that* $\Gamma \vdash \phi : \mathsf{Prop}$*.*

[2] $|\Sigma|$ and $|\Sigma|^\star$ are viewed here as discrete categories.

Then, we can introduce the rules for the logical judgments of the form $\Gamma \mid \Phi \vdash \phi$, where Φ is a finite set of propositions well-formed in Γ.

▶ **Definition 4.5.** *We define four rules for the well-formed formulae previously defined:*

- *$\mathcal{C}$'s rules:*

$$\frac{\Gamma \mid \Phi \vdash \psi}{\Gamma \mid \Phi \vdash \mathcal{C}(\psi)}\ C\text{L-1} \qquad \frac{\Gamma \mid \Phi, \psi \vdash \phi}{\Gamma \mid \Phi, \mathcal{C}(\psi) \vdash \mathcal{C}(\phi)}\ C\text{L-2}$$

- *$\mathcal{U}$'s rules*

$$\frac{\Gamma \mid \Phi, \varphi \vdash \phi \qquad \Gamma \mid \Phi, \mathcal{C}(\varphi), \neg\phi \vdash \psi}{\Gamma \mid \Phi, \varphi \vdash \phi \mathcal{U} \psi}\ \mathcal{U}\text{-}I \qquad \frac{\{\Gamma \mid \Phi, \varphi \vdash \theta \mid \varphi \in \mathsf{u}_{(\Gamma,\Phi)}(\phi, \psi)\}}{\Gamma \mid \Phi, \phi \mathcal{U} \psi \vdash \theta}\ \mathcal{U}\text{-}E$$

where

$$\mathsf{u}_{(\Gamma,\Phi)}(\phi, \psi) := \{\varphi \text{ such that } \Gamma \vdash \varphi : \mathsf{Prop}, \Gamma \mid \Phi, \varphi \vdash \phi, \Gamma \mid \Phi, \mathcal{C}(\varphi), \neg\varphi \vdash \psi\}$$

The Propositional Logic for Closure Operators on Σ *(PLCO) is given by the usual propositional rules (i.e., without the quantifiers) for the typed (intuitionistic) sequent calculus (see e.g. [23]), extended with the four rules above.*

The First Order Logic for Closure Operators on Σ *(FOLCO) is given by the four rules above added to the usual rules for first order logic. Similarily with equality.*

Derivability of sequents is defined in the usual way [38].

▶ Remark 4.6. PLCO corresponds to the Spatial Logic for Closure Spaces considered in [10].

▶ Remark 4.7. Rules $\mathcal{U}$-I and $\mathcal{U}$-E come from the intended meaning of $\phi \mathcal{U} \psi$. In fact, this formula must be interpreted as the "largest region" for which there is no escape from ϕ without passing through ψ.

▶ Remark 4.8. Notice that $\mathcal{U}$-E is an *infinitary* rule saying that a formula θ can be derived from $\phi \mathcal{U} \psi$ if it can be derived from *all* the formulae φ satisfying precise conditions. Thus, this rule shows the second-order nature of the $\mathcal{U}$ operator.

4.2 Categorical semantics of closure logics

In this section we provide a sound and complete categorical semantics of the logics for the closure operators defined above.

▶ **Definition 4.9.** *Two formulae $\phi, \psi \in \mathbf{Form}_\Sigma(\Gamma)$ are* provably equivalent *if $\Gamma \mid \psi \vdash \phi$ and $\Gamma \mid \phi \vdash \psi$. We will denote the quotient of $\mathbf{Form}_\Sigma(\Gamma)$ by this relation with $\mathcal{L}(\Sigma)(\Gamma)$, $[\phi]$ will denote the class of ϕ in it.*

▶ **Proposition 4.10.** *For any signature Σ the following are true:*

1. *$\mathcal{L}(\Sigma)(\Gamma)$ equipped with the order $[\phi] \leq [\psi]$ if and only if $\Gamma \mid \phi \vdash \psi$ is derivable is:*
   - *a meet semilattice in the case we are considering regular logic;*
   - *a Heyting algebra if we are considering propositional or first order logic;*
2. *$[\phi \mathcal{U} \psi]$ is the supremum of the set*

$$\mathsf{u}_\Gamma(\phi, \psi) := \{[\varphi] \in \mathcal{L}(\Sigma)(\Gamma) \text{ such that } \Gamma \mid \varphi \vdash \phi, \Gamma \mid \mathcal{C}(\varphi), \neg\varphi \vdash \psi\}$$

3. *there exists a (elementary) closure or existential doctrine or a (elementary) hyperdoctrine $(\mathcal{L}(\Sigma), \mathfrak{c}_\Sigma)$ on $\mathbf{Cl}(\Sigma)$ sending Γ to $\mathcal{L}(\Sigma)(\Gamma)$.*

Proof.

1. The logical connectives induce a Heyting algebra or a meet semilattice structure on $\mathcal{L}(\Sigma)(\Gamma)$ which has precisely $\leq$ as associated order.
2. From $\mathcal{U}$-I follows that $[\phi \mathcal{U} \psi]$ is an upper bound for u_Γ while $\mathcal{U}$-E implies that $[\phi \mathcal{U} \psi]$ is the least of them.
3. For any morphism $(T_1, ..., T_n) : \Gamma \to \Delta$ substitution of terms gives us a morphism of Heyting algebras/meet semilattices $\mathcal{L}(\Sigma)(\Delta) \to \mathcal{L}(\Sigma)(\Gamma)$; quantifiers gives us the existential doctrine/hyperdoctrine structure (cfr. [38] for the details). In any case have to define a preclosure operator $\mathfrak{c}_{\Sigma,\Gamma}$ on each $\mathcal{L}(\Sigma)(\Gamma)$ but this is easily done defining

$$\mathfrak{c}_{\Sigma,\Gamma} : \mathcal{L}(\Sigma)(\Gamma) \to \mathcal{L}(\Sigma)(\Gamma) \qquad [\phi] \mapsto [\mathcal{C}(\phi)]$$

The $\mathcal{C}$'s rules assure us that $\mathfrak{c}_\Sigma$ is well defined, inflationary and monotone, while an easy induction shows that $\mathcal{L}(\Sigma)_{(T_1,...,T_n)}([\mathcal{C}(\phi)]) = \mathfrak{c}_{\Sigma,\Gamma}(\mathcal{L}(\Sigma)_{(T_1,...,T_n)}(\phi))$ for any $(T_1, ..., T_n) : \Gamma \to \Delta$. We can add fibered equalities, given $\Gamma := [x_i : \sigma_i]$ putting:

$$\delta_{\Gamma\times\Gamma} := \bigwedge_{i=1}^{n} [x_i =_{\sigma_i} y_i]$$

where $\{y_i\}_{i=1}^n$ is a set of fresh variables such that $y_i : \sigma_i$ for any i. ◀

Let us prove the soundness and completeness of the categorical semantics wrt. the various logical fragments.

▶ **Definition 4.11.** *Let $(\mathcal{P}, \mathfrak{c}) : \mathbf{C}^{op} \to \mathbf{InfSL}$ be an (elementary) closure doctrine (existential doctrine/hyperdoctrine) then a morphism of* **cPD** *(***cED***,* **cEED***,* **cEHD***,* **cHD***)* $(\mathcal{M}, \mu) : (\mathcal{L}(\Sigma), \mathcal{C}) \to (\mathcal{P}\mathfrak{c})$ *is a* model of the propositional (first-order) logic (with equality) of closure operators in $(\mathcal{P}, \mathfrak{c})$ *if it is open.*

A sequent $\Gamma \mid \Phi \vdash \psi$ is satisfied by $(\mathcal{M}, \mu)$ *if*

$$\bigwedge_{\phi\in\Phi} \mu_\Gamma(\phi) \leq \mu_\Gamma(\psi)$$

▶ **Theorem 4.12.** *A sequent $\Gamma \mid \Phi \vdash \psi$ is satisfied by the* generic model $(1_{\mathbf{Cl}(\Sigma)}, 1_{\mathcal{L}(\Sigma)})$ *if and only if it is derivable.*

Proof. By definition, $\Gamma \mid \Phi \vdash \psi$ is satisfied if and only if $\bigwedge_{\phi\in\Phi}[\phi] \leq [\psi]$ in $\mathcal{L}(\Sigma)(\Gamma)$, but this is equivalent to the derivability of $\Gamma \mid \bigwedge_{\phi\in\Phi} \phi \vdash \psi$ which in turn is equivalent (applying the conjunction rules a finite number of times) to the derivability of $\Gamma \mid \Phi \vdash \psi$ and we are done. ◀

▶ **Corollary 4.13.** *The above defined categorical semantics for PLCO or FOLCO (with or without equality) is sound and complete.*

Proof. The only thing left to show is soundness for an arbitrary $(\mathcal{P}, \mathfrak{c})$ but this follows at once since each component μ_Γ of μ is monotone. ◀

4.3 Approximating $\mathcal{U}$ in continuous models

As we have remarked before, the rule $\mathcal{U}$-E for the operator $\mathcal{U}$ is infinitary. Although in general this is needed, in this section we will define a class of hyperdoctrines in which the semantics of $\mathcal{U}$ can be given as a supremum of approximants.

▶ **Definition 4.14.** *Let $(\mathcal{P}, \mathfrak{c}) : \mathbf{C}^{op} \to \mathbf{InfSL}$ be a closure doctrine that factors through the category of Heyting algebras. For any object C define the* external boundary*:*

$$\partial^+_C : \mathcal{P}(C) \to \mathcal{P}(C) \qquad \alpha \mapsto \mathfrak{c}_C(\alpha) \wedge \neg\alpha$$

For ϕ and $\psi \in \mathcal{P}(C)$, we define $\phi \mathfrak{U}_C \psi \in \mathcal{P}(C)$ as the supremum, if it exists, of the set

$$\mathfrak{u}_C(\phi, \psi) := \{\varphi \in \mathcal{P}(C) \mid \varphi \leq \phi \text{ and } \partial^+_C(\varphi) \leq \psi\}$$

▶ Remark 4.15. If $\mathcal{P}$ is $\mathcal{L}(\Sigma)$ then $[\phi]\mathfrak{U}_\Gamma[\psi] = [\phi \mathcal{U} \psi]$ for any $[\phi]$ and $[\psi] \in \mathcal{L}(\Sigma)(\Gamma)$.

▶ Remark 4.16. If $(\mathcal{M}, \mu)$ is a model then $\mu_\Gamma(\mathsf{u}_\Gamma(\phi, \psi)) \subset \mathfrak{u}_{\mathcal{M}(\Gamma)}(\mu_\Gamma([\phi]), \mu_\Gamma([\psi]))$ for any Γ.

▶ **Example 4.17.** Let $(X, \mathfrak{c})$ be a pretopological space and S, $T \in \mathcal{P}^p(X, \mathfrak{c})$, then

$$S\mathfrak{U}_{(X,\mathfrak{c})}T = \bigcup\{W \subset S \mid \partial^+_{(X,\mathfrak{c})}(W) \subset T\}$$

i.e. $x \in S\mathfrak{U}_{(X,\mathfrak{c})}T$ if and only if there exists $W \subset S$ such that $X \in W$ and $\partial^+_{(X,\mathfrak{c})}(W) \subset T$.

▶ **Example 4.18.** Let us consider the closure operator $\mathfrak{c}_\epsilon$ on $\mathbf{Set}(-, [0, 1])$ (see Section 3.4). For any $f : X \to [0, 1]$, it is $(\neg f)(x) = 1$ if and only if $f(x) = 0$. So,

$$(\mathfrak{c}_{X,\epsilon}(f) \wedge \neg f)(x) = \begin{cases} \epsilon & f(x) = 0 \\ 0 & f(x) \neq 0 \end{cases},$$

hence, given $g, h : X \to [0, 1]$, $f \in \mathsf{u}_\Gamma(g, h)$ if and only if $f \leq g$ and $h(x) \geq \epsilon$ for any $x \in f^{-1}(0)$.

▶ Remark 4.19. If $(\mathcal{M}, \mu)$ is a model then for any $[\varphi] \in \mathcal{L}(\Sigma)(\Gamma)$ such that $\varphi \in \mathsf{u}_\Gamma(\phi, \psi)$ we have $\mu_\Gamma([\varphi]) \leq \mu_\Gamma([\phi \mathcal{U} \psi])$.

▶ **Definition 4.20.** *Let $(\mathcal{P}, \mathfrak{c})$ be as in Definition 4.14. A model $(\mathcal{M}, \mu) : \mathcal{L}(\Sigma) \to (\mathcal{P}, \mathfrak{c})$ is said* continuous *if the equality*

$$\mu_\Gamma([\phi \mathcal{U} \psi]) = \mu_\Gamma([\phi])\mathfrak{U}_{\mathcal{M}(\Gamma)}\mu_\Gamma([\psi])$$

holds for any context Γ and $[\phi], [\psi] \in \mathcal{L}(\Sigma)(\Gamma)$.

▶ **Proposition 4.21.** *Let Σ be a signature and $(\mathcal{P}, \mathfrak{c})$ a complete (elementary, existential, or hyper)doctrine, i.e. $\mathcal{P}(C)$ is complete for any object C of $\mathbf{C}$; then, for any product preserving functor: $\mathcal{M} : \mathbf{Cl}(\Sigma) \to \mathbf{C}$ and functions*

$$\mu^*_\Gamma : \Pi(\sigma_1, ..., \sigma_n) \to \mathcal{P}(\mathcal{M}(\Gamma))$$

for all $\Gamma = [x_i : \sigma_i]_{i=1}^n$, there exists a unique continuous model $(\mathcal{M}, \mu)$ in $(\mathcal{P}, \mathfrak{c})$ such that

$$\mu_\Gamma([P(x_1, ..., x_n)]) = \mu^*_\Gamma(P)$$

Proof. By induction over n. ◀

▶ **Example 4.22.** Let $\mathcal{X} = \{(X_i, \mathfrak{c}_i)\}_{i \in I}$ be a small family of pretopological spaces and let us define Σ as follows:

$$|\Sigma| := \mathcal{X} \qquad \Gamma(((X_{i_1}, c_{i_1}), ..., (X_{i_n}, c_{i_n})), (X_j, c_j)) := \mathbf{PrTop}(\prod_{k=1}^{n}(X_{i_k}, c_{i_k}), (X_j, c_j))$$

$$\Pi((X_{i_1}, c_{i_1}), ..., (X_{i_n}, c_{i_n})) := \mathcal{P}(\prod_{k=1}^{n} X_{i_k})$$

We can take as $\mathcal{M}$ the unique product preserving functor $\mathbf{Cl}(\Sigma) \to \mathbf{PrTop}$ sending contexts to products and lists of terms to the corresponding product arrow. We can define μ^* sending each predicate $P : (X_{i_1}, \mathfrak{c}_{i_1}), ..., (X_{i_n}, \mathfrak{c}_{i_n})$ to corresponding subset of $\prod_{k=1}^{n} (X_{i_k}, \mathfrak{c}_{i_k})$. Example 4.17 guarantees that this semantics is the same as the one developed in [10].

▶ **Proposition 4.23.** *For any signature Σ a sequent is derivable if and only if it is satisfied by any continuous model.*

Proof. This follows from the fact that the generic model is continuous. ◀

5 Conclusions and future work

In this paper we have introduced *closure (hyper)doctrines* as a theoretical framework for studying the logical aspects of closure spaces. First we have shown the generality of this notion with a range of examples arising naturally from topological spaces, fuzzy sets, algebraic structures, coalgebras, and covering at once also known cases such as Kripke frames and probabilistic frames. Then, we have applied this framework to provide axiomatisations and sound and complete categorical semantics for various fragments of a logic for closure doctrines. In particular, the propositional fragment corresponds to the Spatial Logic for Closure Spaces [10], a modal logic for the specification and verification on spatial properties over preclosure spaces. But the flexibility of our approach allows us to readily obtain closure logics for a wide range of cases (including all the examples presented above). A possible extension is given by *closure triposes* [39]. Triposes are the categorical setting for higher order logic, so these would provide the categorical setting for higher order logic for closure spaces.

Albeit already quite general, the theory presented in this paper paves the way for several extensions. Due to lack of space, we have not been able to present the constructions for modeling logical operators concerning *surroundedness*. To this end, we need to endow doctrines with an object representing the "type of paths"; for more details we refer to the extended version of this work [9].

We can enrich the logic with other spatial modalities, e.g., the spatial counterparts of the various temporal modalities of CTL* [17]. It could be interesting to investigate a spatial logic with fixed points *a la* μ-calculus; to interpret such a logic, we could consider closure hyperdoctrines over Löb algebras [13]. Moreover, it would be interesting to develop some "generic" model checking algorithm for spatial logic. The abstraction provided by the categorical approach can guide the generalization of existing model checking algorithms, such as [10], and suggest new proof methodologies and minimisation techniques.

On a different direction, we are interested in the type theory induced by closure hyperdoctrines. A Curry-Howard isomorphism would yield a functional programming language with constructors for spatial aspects, which would be very useful in *collective spatial programming*, e.g. for collective adaptive systems.

References

1 Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors. *Handbook of Spatial Logics*. Springer, 2007.

2 Michael Atiyah. *Introduction to commutative algebra*. CRC Press, 2018.

3 Tom Avery. Codensity and the Giry monad. *Journal of Pure and Applied Algebra*, 220(3):1229–1251, 2016.

4 Giorgio Bacci and Marino Miculan. Structural operational semantics for continuous state stochastic transition systems. *Journal of Computer and System Sciences*, 81(5):834–858, 2015.

5 Gina Belmonte, Vincenzo Ciancia, Diego Latella, and Mieke Massink. Innovating medical image analysis via spatial logics. In *From Software Engineering to Formal Methods and Tools, and Back*, volume 11865 of *Lecture Notes in Computer Science*, pages 85–109. Springer, 2019.

6 Bodil Biering. *Dialectica interpretations: a categorical analysis*. PhD thesis, IT University of Copenhagen, 2008.

7 Francis Borceux. *Handbook of categorical algebra: volume 1, Basic category theory*, volume 1. Cambridge University Press, 1994.

8 Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proc. POPL*, pages 365–377. ACM, 2000.

9 Davide Castelnovo and Marino Miculan. Closure hyperdoctrines, with paths. *arXiv preprint arXiv:2007.04213*, 2020.

10 Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Specifying and verifying properties of space. In *IFIP International Conference on Theoretical Computer Science*, pages 222–235. Springer, 2014.

11 Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Spatial logic and spatial model checking for closure spaces. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 156–201. Springer, 2016.

12 Djordje Čubrić. On the semantics of the universal quantifier. *Annals of Pure and Applied Logic*, 87(3):209–239, 1997.

13 Pietro Di Gianantonio and Marino Miculan. Unifying recursive and co-recursive definitions in sheaf categories. In *Proc. FoSSaCS*, volume 2987 of *Lecture Notes in Computer Science*, pages 136–150. Springer, 2004.

14 Dikran Dikranjan and Walter Tholen. *Categorical structure of closure operators: with applications to topology, algebra and discrete mathematics*, volume 346. Springer, 2013.

15 Georgi Dimov and Dimiter Vakarelov. Contact algebras and region-based theory of space: a proximity approach i. *Fundamenta Informaticae*, 74(2, 3):209–249, 2006.

16 Georgi Dimov and Dimiter Vakarelov. Contact algebras and region-based theory of space: proximity approach ii. *Fundamenta Informaticae*, 74(2, 3):251–282, 2006.

17 E Allen Emerson and Joseph Y Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

18 Antony Galton. A generalized topological view of motion in discrete space. *Theoretical Computer Science*, 305(1-3):111–134, 2003.

19 Michèle Giry. A categorical approach to probability theory. In *Categorical aspects of topology and analysis*, pages 68–85. Springer, 1982.

20 John Walker Gray. *Formal category theory: adjointness for 2-categories*, volume 391. Springer, 2006.

21 H. Peter Gumm and Tobias Schröder. Products of coalgebras. *Algebra Universalis*, 46:163–185, 2001.

22 Jesse Hughes and Bart Jacobs. Factorization systems and fibrations: Toward a fibred birkhoff variety theorem. *Electronic Notes in Theoretical Computer Science*, 69:156–182, 2003.

23 Bart Jacobs. *Categorical logic and type theory*, volume 141. Elsevier, 1999.

24 Bart Jacobs. *Introduction to Coalgebra*, volume 59. Cambridge University Press, 2017.

25 Bart Jacobs and Ana Sokolova. Exemplaric expressivity of modal logics. *Journal of logic and computation*, 20(5):1041–1068, 2010.

26 Peter T. Johnstone. *Sketches of an elephant: A topos theory compendium*, volume 2. Oxford University Press, 2002.

27 Gregory Maxwell Kelly. A note on relations relative to a factorization system. In *Category Theory*, pages 249–261. Springer, 1991.

28 Anders Kock and Gonzalo E. Reyes. Doctrines in categorical logic. In *Studies in Logic and the Foundations of Mathematics*, volume 90, pages 283–313. Elsevier, 1977.

29 Clemens Kupke and Dirk Pattinson. Coalgebraic semantics of modal logics: an overview. *Theoretical Computer Science*, 412(38):5070–5094, 2011.

30 F. William Lawvere. Adjointness in foundations. *Dialectica*, 23(3-4):281–296, 1969.

31 F. William Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. *Applications of Categorical Algebra*, 17:1–14, 1970.

32 Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory*. Springer Science & Business Media, 2012.

33 Maria Emilia Maietti, Fabio Pasquali, and Giuseppe Rosolini. Triposes, exact completions, and Hilbert's ε-operator. *Tbilisi Mathematical Journal*, 10(3):141–166, 2017.

34 Maria Emilia Maietti and Giuseppe Rosolini. Quotient completion for the foundation of constructive mathematics. *Logica Universalis*, 7(3):371–402, 2013.

35 Michael Makkai and Gonzalo E. Reyes. *First order categorical logic: model-theoretical methods in the theory of topoi and related categories*, volume 611. Springer, 2006.

36 Marino Miculan and Giorgio Bacci. Modal logics for brane calculus. In *Proc. CMSB*, volume 4210 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2006.

37 Fabio Pasquali. A co-free construction for elementary doctrines. *Applied Categorical Structures*, 23(1):29–41, 2015.

38 Andrew M. Pitts. Categorical logic. Technical report, University of Cambridge, 1995.

39 Andrew M Pitts. Tripos theory in retrospect. *Mathematical structures in computer science*, 12(3):265–279, 2002.

40 Robert A. G. Seely. Hyperdoctrines, natural deduction and the Beck condition. *Mathematical Logic Quarterly*, 29(10):505–542, 1983.

41 Oswald Wyler. *Lecture notes on topoi and quasitopoi*. World Scientific, 1991.

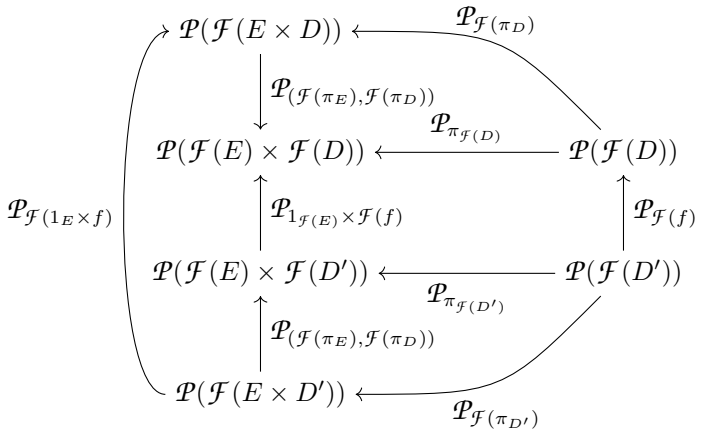
42 Lotfi A. Zadeh. Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific, 1996.

A Omitted proofs

Proof of Proposition 2.7. We have to show that $\mathcal{P}_{\mathcal{F}(\pi_D)}$ has a left adjoint for any projection $\pi_D : E \times D \to D$ but this follows at once since the diagonal arrow in the diagram

$$\begin{array}{ccc} \mathcal{P}(\mathcal{F}(D)) & \xrightarrow{\mathcal{P}_{\pi_{\mathcal{F}(D)}}} & \mathcal{P}(\mathcal{F}(E) \times \mathcal{F}(D)) \\ {\scriptstyle \mathcal{P}_{\mathcal{F}(\pi_D)}}\downarrow & \nearrow {\scriptstyle \mathcal{P}_{(\mathcal{F}(\pi_E),\mathcal{F}(\pi_D))}} & \\ \mathcal{P}(\mathcal{F}(E \times D)) & & \end{array}$$

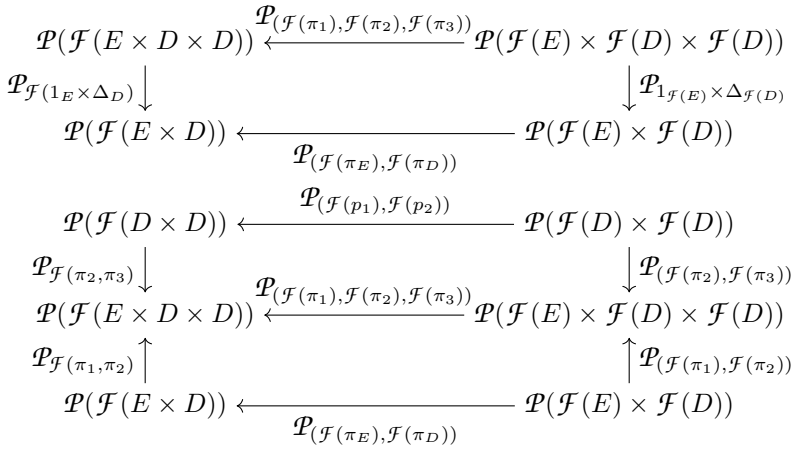
is an isomorphism, hence we can define $\exists_{\pi_D}$ as the composition $\exists_{\pi_{\mathcal{F}(D)}} \circ \mathcal{P}_{(\mathcal{F}(\pi_E),\mathcal{F}(\pi_D))}$. The same argument shows that $\forall_{\pi_{\mathcal{F}(D)}} \circ \mathcal{P}_{(\mathcal{F}(\pi_E),\mathcal{F}(\pi_D))}$ is the right adjoint to $\mathcal{P}_{\mathcal{F}(\pi_D)}$ whenever $\forall_{\pi_{\mathcal{F}(D)}}$ exists. Let now $f : D' \to D$ be an arrow in $\mathbf{D}$, the two Beck-Chevalley conditions follow from the commutativity of



and the fact that both the upper and the lower vertical arrow are isomorphisms since $\mathcal{F}$ preserves products. For Frobenius reciprocity:

$$\begin{aligned}&\exists_{\mathcal{F}(\pi_D)}(\mathcal{P}_{\mathcal{F}(\pi_D)}(\alpha) \wedge \beta) = \exists_{\pi_{\mathcal{F}(D)}}(\mathcal{P}_{(\mathcal{F}(\pi_E),\mathcal{F}(\pi_D))}(\mathcal{P}_{\mathcal{F}(\pi_D)}(\alpha) \wedge \beta))\\ &= \exists_{\pi_{\mathcal{F}(D)}}(\mathcal{P}_{\pi_{\mathcal{F}(D)}}(\alpha) \wedge \mathcal{P}_{(\mathcal{F}(\pi_E),\mathcal{F}(\pi_D))}(\beta)) = \alpha \wedge \exists_{\pi_{\mathcal{F}(D)}}(\mathcal{P}_{\mathcal{F}(\pi_D)}(\alpha) \wedge \beta) = \alpha \wedge \exists_{\mathcal{F}(\pi_D)}(\beta)\end{aligned}$$

So we're left with the fibered equalities, but the commutativity of



entails that $\mathcal{P}_{(\mathcal{F}(p_1),\mathcal{F}(p_2))}(\delta_{\mathcal{F}(D)})$ has the property of a fibered equality. ◀

Proof of Proposition 2.8. (Cfr. [23, 31] and lemma 1.5.8 of [26], vol. 1 for the hyperdoctrine case). It is enough to define

$$\exists_f(\alpha) := \exists_{\pi_D}(\mathcal{P}_{f \times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\alpha)) \qquad \forall_f(\alpha) := \forall_{\pi_D}(\mathcal{P}_{f \times 1_D}(\delta_D) \to \mathcal{P}_{\pi_C}(\alpha))$$

Let us now show that $\exists_f \dashv \mathcal{P}_f$.

If $\exists_f(\alpha) \leq \beta$	If $\alpha \leq \mathcal{P}_f(\beta)$
$\alpha = \alpha \wedge \top_C = \alpha \wedge \exists_{\pi_2}(\delta_C)$	$\exists_f(\alpha) \leq \exists_f(\mathcal{P}_f(\beta))$
$\leq \alpha \wedge \exists_{\pi_2}(\mathcal{P}_{f\times f}(\delta_D))$	$= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\mathcal{P}_f(\beta))$
$= \exists_{\pi_2}(\mathcal{P}_{f\times f}(\delta_D) \wedge \mathcal{P}_{p_2}(\alpha))$	$= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{1_D\times f}(\mathcal{P}_{q_2}(\beta)))$
$= \exists_{\pi_2}(\mathcal{P}_{1_C\times f}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\alpha))$	$= \exists_{\pi_D}(\mathcal{P}_{1_D\times f}(\delta_D) \wedge \mathcal{P}_{1_D\times f}(\mathcal{P}_{q_2}(\beta)))$
$= \mathcal{P}_f(\exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\alpha)))$	$= \exists_{\pi_D}(\mathcal{P}_{1_D\times f}(\delta_D \wedge \mathcal{P}_{q_2}(\beta))) = \exists_{\pi_D}(\mathcal{P}_{1_D\times f}(\exists_{\Delta_D}(\beta)))$
$= \mathcal{P}_f(\exists_f(\alpha)) \leq \mathcal{P}_f(\beta)$	$\leq \exists_{\pi_D}(\mathcal{P}_{1_D\times f}(\mathcal{P}_{q_1}(\beta))) = \exists_{\pi_D}(\mathcal{P}_{\pi_D}(\beta)) \leq \beta$

where p_2 is the second projection $C \times C \to C$ and q_1 and q_2 those $D \times D \to D$. For the adjunction $\mathcal{P}_f \dashv \forall_f$ we already know that $\exists_{\pi_C} \dashv \mathcal{P}_{\pi_C}$, $\mathcal{P}_{f\times 1_D}(\delta_D) \wedge (-) \dashv \mathcal{P}_{f\times 1_D}(\delta_D) \to (-)$ and $\mathcal{P}_{\pi_D} \dashv \forall_{\pi_D}$, so it is enough to show that $\mathcal{P}_f(\beta) = \exists_{\pi_C}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_D}(\beta))$ for all $\beta \in \mathcal{P}(D)$. But this is easily done:

$$\begin{aligned}&\exists_{\pi_C}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_D}(\beta)) = \exists_{\pi_C}(\mathcal{P}_{1_D\times f}(\delta_D) \wedge \mathcal{P}_{1_D\times f}(\mathcal{P}_{\pi_1}(\beta)))\\ &= \exists_{\pi_C}(\mathcal{P}_{1_D\times f}(\delta_D \wedge \mathcal{P}_{\pi_1}(\beta))) = \mathcal{P}_f(\exists_{\pi_2}(\exists_{\Delta_D}(\beta))) = \mathcal{P}_f(\beta)\end{aligned}$$

Where π_2 is the second projection $D \times D \to D$. We're left with Frobenius reciprocity: the inequality $\exists_f(\mathcal{P}_f(\beta) \wedge \alpha) \leq \beta \wedge \exists_f(\alpha)$ follows from adjointness, let's show the other. If π_1 and π_2 are the projections from $D \times D$, then

$$\begin{aligned}&\exists_f(\mathcal{P}_f(\beta) \wedge \alpha) = \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\mathcal{P}_f(\beta) \wedge \alpha))\\ &= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\mathcal{P}_f(\beta)) \wedge \mathcal{P}_{\pi_C}(\alpha)) = \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{f\times 1_D}(\mathcal{P}_{\pi_1}(\beta)) \wedge \mathcal{P}_{\pi_C}(\alpha))\\ &= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D \wedge \mathcal{P}_{\pi_1}(\beta)) \wedge \mathcal{P}_{\pi_C}(\alpha)) \leq \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D \wedge \mathcal{P}_{\pi_2}(\beta)) \wedge \mathcal{P}_{\pi_C}(\alpha))\\ &= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{f\times 1_D}(\mathcal{P}_{\pi_2}(\beta)) \wedge \mathcal{P}_{\pi_C}(\alpha)) = \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_D}(\beta) \wedge \mathcal{P}_{\pi_C}(\alpha))\\ &= \exists_{\pi_D}(\mathcal{P}_{f\times 1_D}(\delta_D) \wedge \mathcal{P}_{\pi_C}(\alpha)) \wedge \beta = \exists_f(\alpha) \wedge \beta\end{aligned}$$

where we have used $\delta_D \wedge \mathcal{P}_{\pi_1}(\beta) \leq \mathcal{P}_{\pi_2}(\beta)$ which follows from the definition of $\exists_{\Delta_D}$. ◀

Proof of Proposition 3.24. We have to show continuity of all arrows $f : (A, \alpha) \to (B, \beta)$. Let $\xi \in (B, \beta)$ and $x \in A$, we have four cases:

1. $f^*(\xi)(x) + \epsilon_{(A,\alpha)}(x) < \alpha(x)$ and $\xi(x) + \epsilon_{(B,\beta)}(x) < \beta(x)$.

$$\begin{aligned}&(\mathfrak{c}^{\mathcal{E}}_{(A,\alpha)}(f^*(\xi)))(x) = (f^*(\xi) + \epsilon_{(A,\alpha)})(x) = (\alpha(x) \wedge \xi(f(x))) + \epsilon_{(A,\alpha)}(x)\\ &= \alpha(x) \wedge (\xi(f(x)) + \epsilon_{(A,\alpha)}(x)) \leq \alpha(x) \wedge (\xi(f(x)) + \epsilon_{(B,\beta)}(f(x))) = f^*(\mathfrak{c}^{\mathcal{E}}_{(B,\beta)}(\xi))(x)\end{aligned}$$

2. $f^*(\xi)(x) + \epsilon_{(A,\alpha)}(x) < \alpha(x)$ and $\xi(f(x)) + \epsilon_{(B,\beta)}(f(x)) \geq \beta(f(x))$. Notice that $\alpha(x) \leq \beta(f(x))$ so $f^*(\mathfrak{c}^{\mathcal{E}}_{(B,\beta)}(\xi))(x) = \alpha(x)$ and thus

$$\begin{aligned}&(\mathfrak{c}^{\mathcal{E}}_{(A,\alpha)}(f^*(\xi)))(x) = (f^*(\xi) + \epsilon_{(A,\alpha)})(x) = (\alpha(x) \wedge \xi(f(x))) + \epsilon_{(A,\alpha)}(x)\\ &= \alpha(x) \wedge (\xi(f(x)) + \epsilon_{(A,\alpha)}(x)) = \alpha(x) = f^*(\mathfrak{c}^{\mathcal{E}}_{(B,\beta)}(\xi))(x)\end{aligned}$$

3. $f^*(\xi)(x) + \epsilon_{(A,\alpha)}(x) \geq \alpha(x)$ and $\xi(x) + \epsilon_{(B,\beta)}(x) < \beta(x)$.

$$\begin{aligned}&(\mathfrak{c}^{\mathcal{E}}_{(A,\alpha)}(f^*(\xi)))(x) = \alpha(x) = \alpha(x) \wedge (\xi(f(x)) + \epsilon_{(A,\alpha)}(x))\\ &\leq \alpha(x) \wedge (\xi(f(x)) + \epsilon_{(B,\beta)}(f(x))) = f^*(\mathfrak{c}^{\mathcal{E}}_{(B,\beta)}(\xi))(x)\end{aligned}$$

4. $f^*(\xi)(x) + \epsilon_{(A,\alpha)}(x) \geq \alpha(x)$ and $\xi(x) + \epsilon_{(B,\beta)}(x) \geq \beta(x)$.

$$(\mathfrak{c}^{\mathcal{E}}_{(A,\alpha)}(f^*(\xi)))(x) = \alpha(x) = \alpha(x) \wedge \beta(f(x)) = f^*(\mathfrak{c}^{\mathcal{E}}_{(B,\beta)}(\xi))(x)$$

We are left with additivity, but this follows immediately since, for ξ and $\zeta \in \mathcal{FzSub}(A,\alpha)$ and $x \in A$, $(\xi \vee \zeta)(x)$ is $\xi(x)$ or $\zeta(x)$. ◀

Proof of Lemma 3.31. 1. Clearly $\alpha \leq \mathsf{pre}_{\gamma_X}(\alpha)$; if $\alpha \leq \beta$ we also have $\mathcal{P}_{\gamma_X}(\Box_X(\alpha)) \leq \mathcal{P}_{\gamma_X}(\Box_X(\beta))$ from which monotonicity follows. Tanke now an arrow f between $\gamma_X : X \to \mathcal{F}(X)$ and $\gamma_Y : Y \to \mathcal{F}(Y)$, thus $\mathcal{F}(f) \circ \gamma_X = \gamma_Y \circ f$ so

$$\begin{aligned}\mathsf{pre}_{\gamma_X}(\mathcal{P}_f(\alpha)) &= \mathcal{P}_f(\alpha) \vee \mathcal{P}_{\gamma_X}(\Box_X(\mathcal{P}_f(\alpha))) = \mathcal{P}_f(\alpha) \vee \mathcal{P}_{\gamma_X}(\mathcal{P}_{\mathcal{F}(f)}(\Box_Y(\alpha)))\\ &= \mathcal{P}_f(\alpha) \vee \mathcal{P}_f(\mathcal{P}_{\gamma_Y}(\Box_Y(\alpha))) = \mathcal{P}_f(\alpha \vee \mathcal{P}_{\gamma_Y}(\Box_Y(\alpha))) = \mathcal{P}_f(\mathsf{pre}_{\gamma_Y}(\alpha))\end{aligned}$$

2. For any $\alpha \in \mathcal{P}(X)$

$$\mathcal{A}_{\gamma_X}(\alpha) := \{\Box_X(\beta) \mid \beta \in \mathcal{N}_{\gamma_X}(\alpha)\} \qquad \mathcal{B}_{\gamma_X}(\alpha) := \{\mathcal{P}_{\gamma_X}(\Box_X(\beta)) \mid \beta \in \mathcal{N}_{\gamma_X}(\alpha)\}$$

then by the hypothesis on $\mathcal{P}_{\gamma_X}$ and the previous point we have

$$\alpha \leq \inf(\mathcal{B}_{\gamma_X}(\alpha)) = \mathcal{P}_{\gamma_X}(\inf(\mathcal{A}_{\gamma_X}(\alpha))) = \mathcal{P}_{\gamma_X}(\Box_X(\mathsf{s}_{\gamma_X}(\alpha)))$$

3. The inequality $\alpha \leq \mathsf{suc}_{\gamma_X}(\alpha)$ follows at once, if $\alpha \leq \beta$ we have $\mathcal{P}_{\gamma_X}(\Box_X(\alpha))$ as in the first point but this implies that $\mathcal{N}_{\gamma_X}(\beta) \subset \mathcal{N}_{\gamma_X}(\alpha)$. Hence, $\bigwedge_{\theta \in \mathcal{N}_{\gamma_X}(\alpha)} \theta \leq \bigwedge_{\theta \in \mathcal{N}_{\gamma_X}(\beta)} \theta$, from which we deduce the monotonicity of suc_{γ_X}. For any morphism $f : \gamma_X \to \gamma_Y$ of coalgebras we have

$$\mathcal{P}_f(\alpha) \leq \mathcal{P}_f(\mathcal{P}_{\gamma_Y}(\Box_Y(\theta))) = \mathcal{P}_{\gamma_X}(\mathcal{P}_{\mathcal{F}(f)}(\Box_Y(\theta))) = \mathcal{P}_{\gamma_X}(\Box_X(\mathcal{P}_f(\theta)))$$

for all $\theta \in \mathcal{N}_Y(\alpha)$. Hence $\mathcal{P}_f(\theta) \in \mathcal{N}_X(\mathcal{P}_f(\alpha))$, then $\mathsf{s}_{\gamma_X}(\mathcal{P}_f(\alpha)) \leq \inf(\mathcal{P}_f(\mathcal{N}_Y(\alpha)))$:

$$\begin{aligned}&\mathsf{suc}_{\gamma_X}(\mathcal{P}_f(\alpha)) = \mathcal{P}_f(\alpha) \vee \mathsf{s}_{\gamma_X}(\mathcal{P}_f(\alpha)) \leq \mathcal{P}_f(\alpha) \vee \inf(\mathcal{P}_f(\mathcal{N}_Y(\alpha)))\\ &\leq \mathcal{P}_f(\alpha) \vee \mathcal{P}_f(\inf(\mathcal{N}_Y(\alpha))) = \mathcal{P}_f(\alpha \vee \mathsf{s}_{\gamma_Y}(\alpha)) = \mathcal{P}_f(\mathsf{suc}_{\gamma_Y}(\alpha))\end{aligned}$$

and we are done. ◀

How to Write a Coequation

Fredrik Dahlqvist
Department of Computer Science, University College London, UK

Todd Schmid
Department of Computer Science, University College London, UK

Abstract

There is a large amount of literature on the topic of covarieties, coequations and coequational specifications, dating back to the early seventies. Nevertheless, coequations have not (yet) emerged as an everyday practical specification formalism for computer scientists. In this review paper, we argue that this is partly due to the multitude of syntaxes for *writing down* coequations, which seems to have led to some confusion about what coequations are and what they are for. By surveying the literature, we identify four types of syntaxes: *coequations-as-corelations*, *coequations-as-predicates*, *coequations-as-equations*, and *coequations-as-modal-formulas*. We present each of these in a tutorial fashion, relate them to each other, and discuss their respective uses.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Theory of computation → Formal languages and automata theory; Theory of computation → Program specifications

Keywords and phrases Coalgebra, coequation, covariety

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.13

Category (Co)algebraic pearls

Funding *Fredrik Dahlqvist*: Leverhulme Project Grant 'Verification of machine learning algorithms'

Acknowledgements The authors are most grateful to Alexander Kurz for his services as history consultant. The responsibility for any mistake or mischaracterisation lie solely with the authors.

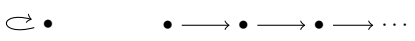
1 Introduction

Characterising algebras by the equations they satisfy is common practice. Equations are simple to write down: they consist of a pair of terms, or elements of an initial algebra. Equations are also simple to interpret: terms denote constructions, so an equation between two terms asserts that two constructions produce equivalent objects. The terms-as-constructions interpretation of equations is prevalent in programming language theory. A programming language is a syntax for denoting programs, so equations state equivalences between programs. Their importance can be seen in λ-calculus [15, 85, 84, 70], process algebra [35], Kleene algebra [86, 58, 24] and its extensions [59, 36, 56, 54, 91, 88], and related areas [94, 75].

A common thread, running through the many examples of equational reasoning in computer science, is that equations can be used to state *behavioural* equivalences between programs. For the purposes of this review, behaviours are what are obtained from dualizing, in the category theoretic sense, the concept of term. That is, a behaviour is an element of a final *coalgebra*. The dual study to algebra, coalgebra, constitutes a whole subfield of computer science dedicated to state-based dynamical systems, the sort of systems that exhibit behaviours [79, 43, 52]. Dualizing algebra not only takes terms to behaviours, but also equations to *coequations*, the main focus of this article.

Broadly, a coequation is a constraint on the dynamics of a state-based system. A coalgebra satisfies a coequation if its dynamics operate within the constraint. This situation is familiar to those working in automata theory, since deterministic automata are examples

of state-based dynamical systems. For a fixed alphabet A, any set of languages $\mathcal{L} \subseteq 2^{A^*}$ determines a coequation satisfied by those automata that only accept languages in $\mathcal{L}$. We call these coequations *behavioural*, as they consist of a set of states in the final automaton $2^{A^*} \to 2 \times (2^{A^*})^A$. Not all coequations are behavioural: following [93, 21] in viewing Kripke frames as coalgebraic dynamical systems for the powerset functor, modal formulas provide illustrative examples of nonbehavioural coequations. For example, reflexivity is a modally definable constraint on the dynamics of frames (witnessed by the modal formula $\Box p \to p$), despite the following two Kripke frames being behaviourally indistinguishable.



Rather, reflexivity is a coequation which requires two *colours* to be stated, p and $\neg p$. The concept of colour (or label) is key to moving beyond purely behavioural specifications, and can be understood as the formal dual to the notion of variable in algebra. Just like the commutativity of a binary operation requires two variables to be stated, the reflexivity of a Kripke frame requires two colours.

Coequations have arguably not seen much use by computer scientists, in spite of a large body of theoretical results. We postulate that one of the main reasons for this is that there is no one universally accepted way to *write down a coequation*. It is not hard to see why: while an equation relates two, finite, tree-structures which can be written-down unambiguously in one dimension with the use of brackets, there is no universal syntax for describing *constraints* on structures which are often *inherently infinitary*. In fact, a variety of syntaxes for writing down coequations have been proposed in the literature, leading to a certain ambiguity surrounding the term coequation, especially since some of them are less expressive than others. This being said, it could equally be argued that coequations are used extensively, if unknowingly, by computer scientists, in the shape of modal logics [18]. For example, languages like Linear Temporal Logic [92, 38, 37] and Computation Tree Logic [34, 22] are efficient syntaxes for specifying coequations.

The purpose of this paper is three-fold. The first is to survey and organise the literature on coequations. The second is to act as a tutorial on coequations and coequational specification. We assume basic knowledge of category theory and focus on Set-coalgebras. Finally, we aim to present a systematic account of what the various notions of coequations are, how they are related to one another, and what role they have to play in theoretical computer science.

The paper is structured as follows. We start with a review of the literature on coequations in § 2, highlighting a number of formalisms for defining and specifying coequations. We group these approaches into four paradigms, which we examine in detail. First, in § 3, we present a notion of coequation which dualizes exactly the notion of equation in Universal Algebra, and which we call *coequation-as-corelation*. Second, we present the view that a coequation is a predicate on a cofree coalgebra. We call this notion *coequation-as-predicate* and discuss it in detail in § 4. Third, we discuss *coequations-as-equations* in § 5 and relate them to coequations-as-corelations. The last paradigm we explore is that of coequations as modal formulas in § 6. Finally, we conclude in § 7 with some thoughts on the uses of each formalism and some recent appearances of coequations in computer science.

2 A brief history of coequations

As far as we are aware, the first mention of the word "coequation"– or more precisely "coequational" – dates back a series of papers by Davis [29, 30, 31, 32, 33] starting in 1970 and focusing on finding examples of comonadic/cotripleable categories. The earliest work that deals with covarieties of coalgebras as we now understand them, seems to be the 1985

paper [71], which presents a category-theoretic account of a dual to Birkhoff's HSP theorem. This work focuses on coalgebras for polynomial functors on the category Set. It describes an unusual approach to dualising Birkhoff's HSP theorem which reduces the problem to the ordinary version of the theorem by turning every coalgebra $X \to FX$ into an algebra $2^{FX} \to 2^X$, and by introducing an infinitary equational logic extending that of complete atomic boolean algebras to define varieties of such algebras. The idea of establishing a bridge between coequations and equations was explored again in [13] and [83] where conditions for a full duality between equations and coequations are given.

A few years after [71], work on the notion of *terminal coalgebra* in [1, 2, 16] laid the ground for [47, 76, 50], which proposed coalgebras as a semantic framework to formalise behaviours in object-oriented programming and infinite data structures such as streams and trees. From the onset, the aim of this line of research was to syntactically specify classes of behaviours and, although the terms "covariety" and "coequation" do not appear in *op.cit.*, a lot of the questions which we will explore in this paper can already be found in Hensel and Reichel's [47] and Jacobs' [50]. Both approaches propose *equational specifications* of coalgebras for polynomial functors, based on the signature of the functor. For example, the equation $\mathsf{head}(\mathsf{tail}(\mathsf{x})) = \mathsf{head}(\mathsf{x})$, where the "destructor" signature $\mathsf{head}, \mathsf{tail}$ can be read off the functor $F(X) = X \times A$, characterises constant streams in the terminal F-coalgebra A^ω. Such equations are called *state equations* in [47] since they must hold at every state, and [50] gives a concrete construction of the class of behaviours satisfying such an equation via the notion of "mongruence" (a terminology which mercifully has not caught on). This kind of coequational specification via equations, which we refer to as *coequations-as-equations*, is also used in Cîrstea's 1999 [20], which is the first full paper to use the term "coequation" in the sense we understand today. Roşu's [78] from 2001 follows the same approach of *coequations-as-equations*. While intuitive, this way of "writing coequations" is limited to endofunctors of a specific shape. Another, more powerful, way of writing coequations-as-equations was developed by Kurz and Rosický in [68] using an equality relation between terms built from a signature, but with different notions of (co)operation and term. In this framework, every covariety over Set can be presented in a coequation-as-equation format.

Jacobs' [50] pioneered this specification format but also asked the following questions, which motivated a lot of the subsequent research on the topic: 1) Can a sound and complete logic to reason about coalgebras be devised? 2) Can a version of Birkhoff's theorems be proved for suitable classes of coalgebras? The first step towards answering these question was taken by Rutten's influential 1996 technical report [79, 80], which introduces the notion of "colours" of a cofree coalgebra, the concept of *covariety*, and the first of many dual versions of Birkhoff's HSP theorem. The term "coequation" does not appear in [79, 80], but it is worth noting that a concrete specification of a covariety is given by a *subcoalgebra of a cofree coalgebra*, in contrast to the equational presentation of [47, 50, 20, 78]. We refer to this type of coequational specification as *coequations-as-predicates*.

An important moment in the history of coequations was the first Workshop on Coalgebraic Methods in Computer Science (CMCS), organised by Jacobs, Moss, Reichel and Rutten and held in Lisbon in 1998. Several papers on covarieties were presented [45, 64, 77], and the next few years saw an explosion of research in this area. In retrospect, CMCS 1998 provided much of the momentum behind the subsequent blossoming of this new field of research.

In their 1998 CMCS paper [45], Gumm and Schröder pick up the study of covarieties defined via a subcoalgebra in [79]. They isolate precisely which subcoalgebras define a covariety and describe the closure properties of covarieties closed under bisimulation, which they call *complete covarieties*. It was subsequently shown in [11, 49] that these covarieties,

more aptly called *behavioural covarieties*, are precisely those which can be described by a coequation-as-predicate over one colour, that is to say by a subcoalgebra of the terminal coalgebra. Coequations as subcoalgebras of a cofree coalgebras (or more abstractly as regular monomorphisms with cofree codomain) and the covarieties they define are also discussed in detail in [6, 7] where a dual to Birkhoff's HSP theorem is given.

Hughes' 2001 thesis [49] presents a very detailed abstract account of the *coequations-as-predicates* perspective. A coequation is no longer required to be defined by a sub*coalgebra*, but can simply be a sub*set* of a cofree coalgebra [49, §3.6.3]. Closure operators defined and studied in [49, 48, 51] connect these two flavours of the *coequations-as-predicates* paradigm, by constructing the (invariant) sub*coalgebra* generated by a sub*set* of behaviours. More abstractly, [49] also considers a coequation as a subcoalgebra of a regular injective coalgebra (e.g. a cofree coalgebra). This additional abstraction dualizes the description of equations as quotients of regular projective algebras due to [14], but introduces subtle differences on the closure properties of covarieties which are discussed by Goldblatt in [39, 23]. In this paper, we only consider subcoalgebras of cofree coalgebras or subsets of their carriers.

A third version of the *coequations-as-predicates* paradigm was proposed by Gumm in [44], where a coequation is a single pattern (i.e. element of a cofree coalgebra), but a pattern that *must be avoided*. In other words, this is a *coequation-as-predicate* defined by the complement of a singleton, i.e. understood as a *pattern avoidance constraint*. This fruitful idea was the source of many interesting examples in [7], and the basis for coequational logics in [4, 89, 90]. These logics are based on two observations which can already be found in [44], namely that if there exists a state x witnessing a pattern f, then any "successor pattern" of f must be witnessed by some state y (namely one of the successors of x). Similarly, if there exists a state x witnessing a pattern f, there must exist a recolouring/relabelling of this state which witnesses a similar relabelling of the pattern f. Adamek [4] shows that these two observations are enough to define a sound and complete coequational system in which new coequations (avoidance patterns) can be deduced from known ones. This logic is most natural for polynomial functors, but can also be made to work for very large class of accessible functors through the notion of functor presentation [4, 89, 90].

Finally, we mention generalisations of *coequations-as-predicates* to the case where cofree coalgebras do not exist. Adámek and Porst generalise coequations-as-predicates by considering regular monomorphisms into *any* element of the cofree coalgebra chain, whether it stabilises or not [7]. Kurz and Rosický in [68] describe the notion of *implicit operations* which permits an equivalent notion of coequation-as-predicate for functors which have no cofree coalgebras. Finally, Adámek describes a comprehensive solution to this problem by considering generalised coequations-as-predicates as subchains of the entire cofree-coalgebra chain in [3].

A related but different notion of coequation was proposed in 2000 by Wolter [95] and Kurz [65], systematically dualizing the picture from categorical Universal Algebra. Since a set of equations can be understood as a relation between terms in a free algebra (categorically a span), Wolter proposes that a coequation should be seen as a *corelation* on the carrier of a cofree coalgebra over some set of colours (categorically, a cospan). Similarly, Kurz proposes to consider a *cocongruence* on the cofree coalgebra. As in the *coequation-as-predicate* paradigm, these two approaches reflect the fact that one may consider a coequation as a structure on the *carrier* of a cofree coalgebra (a corelation), or on the cofree coalgebra itself (a cocongruence). We will refer to this approach as *coequations-as-corelations*. This approach neatly dualizes the well-known theory of equations, but the notion of corelation is not very intuitive, as pointed out by Hughes [49]. Nevertheless, we hope to provide some intuition in § 3 and § 5.

At the same period Kurz also proposed *modal logic* as a language for specifying covarieties [63, 65, 66, 67]. This is our last paradigm for coequations: *coequations-as-modal-formulas*. Following our discussion in the introduction, we know that for finitely branching Kripke frames there exists a *final* Kripke frame coloured by (sets of) propositional variables, and any modal formula ϕ selects the states in this Kripke frame in which ϕ is valid, i.e. defines a predicate on a cofree coalgebra. A modal formula can thus be seen as a syntax for *coequations-as-predicates*. However, these are very particular predicates: they have a simple and intuitive syntax, access to a countable set of colours (the propositional variables), and to 1-step ahead colours (through modalities). This idea can easily be extended to modal logics for polynomial functors [66]. However, the idea of modal logic as a specification language for covarieties found in [63, 65, 66, 67] was in some way too prescient: the appropriate extension of modal logic – *coalgebraic modal logic* – was still in its infancy. Moss' logic [73] had only just been published, and neither the predicate lifting formalism of Pattinson [74] nor the abstract formalism of Kupke, Kurz, and Pattinson [60, 61, 53] had been developed. As a consequence, Kurz's insight of *coequations-as-modal-formulas* was only worked out for standard modal logics [66]. His abstract notion of *modal predicate* [65] – where the term "modal" is meant as "invariant under bisimulation" – is not based on a particular syntax, but is defined as a monomorphism into a cofree coalgebra, i.e. as a coequation-as-predicate.

3 Coequations-as-corelations

Coequations-as-corelations is the notion of coequation which most faithfully dualizes the notion of equation from Universal Algebra. It is not the simplest approach, but it exposes the underlying machinery in its entirety. Syntactically, it is a formalism that resembles equations because it uses a *pairs of expressions*. However, whilst an equation between a pair of expressions *forces* an equality to be witnessed via a quotient, a *coequation-as-corelation* involves a pair of expressions which *selects* an "existing" equality via an equaliser. Much of the material in this section can be found in [95, 65, 11, 48, 49, 25]. We present the classical picture from Universal Algebra in §3.1, and then dualize it in §3.2.

3.1 Equations, relations and varieties of algebras

We will not go to the level of generality of [11, 49, 48] but instead focus on algebras for Set-endofunctors. Let $T : \mathsf{Set} \to \mathsf{Set}$ be an endofunctor and let $\mathsf{Alg}(T)$ denote the category of T-algebras and T-algebra morphisms. There exists an obvious forgetful functor $U_T : \mathsf{Alg}(T) \to \mathsf{Set}$ which keeps the carrier and forgets the algebraic structure. A functor T is called a *varietor* [9] if this functor has a left-adjoint $F_T : \mathsf{Set} \to \mathsf{Alg}(T)$ which builds free T-algebras over any given set of variables. We will drop the subscripts and simply write $F \dashv U$ if this causes no ambiguity. It follows from the adjunction that any map $h : X \to U(A, \alpha)$ can be freely extended to a T-algebra morphism $\hat{h} : FX \to (A, \alpha)$ explicitly constructed as $\hat{h} \triangleq \varepsilon^T_{(A,\alpha)} \circ Fh$, where ε^T is the counit of the adjunction.

For any varietor T we define a *set of T-equations over a set of variables X* is a pair of arrows $e_1, e_2 : E \rightrightarrows UFX$. A set of equations is thus represented as a *span*, the categorical embodiment of the notion of relation.

A T-algebra (A, α) *satisfies a set of equations* $e_1, e_2 : E \rightrightarrows UFX$ if for all *valuations* $v : X \to U(A, \alpha)$, $U\hat{v} \circ e_1 = U\hat{v} \circ e_2$, i.e. if the map $\hat{v}$ which recursively computes the interpretation in (A, α) of formal terms from FX, returns the same output for the left- and right-hand-side of each equation in E. This can be rephrased as a universal property in $\mathsf{Alg}(T)$ by saying that (A, α) satisfies a set of equations $e_1, e_2 : E \rightrightarrows UFX$ if any T-algebra

morphism $f : FX \to (A, \alpha)$ factors uniquely through the coequalizer q of $\hat{e}_1, \hat{e}_2$.

$$\begin{array}{ccccc} FE & \overset{\hat{e}_1}{\underset{\hat{e}_2}{\rightrightarrows}} & FX & \xrightarrow{\quad q \quad} & (Q, \nu) \\ & & \downarrow{\scriptstyle f} & \swarrow & \\ & & (A, \alpha) & & \end{array} \tag{1}$$

Since any morphism $f : FX \to (A, \alpha)$ is of the shape $f = \hat{v}$ for some $v : X \to U(A, \alpha)$, we recover the standard notion of equation satisfaction. An object (A, α) with the universal property in (1) is said to be *orthogonal* to $q : FX \twoheadrightarrow (Q, \nu)$, written $q \perp (M, \alpha)$.

The *variety of T-algebras defined by the set of T-equations* $e_1, e_2 : E \rightrightarrows UFX$ is defined as the class of all T-algebras which are orthogonal to the coequalizer of the adjoint morphisms $\hat{e}_1, \hat{e}_2 : FE \rightrightarrows FX$, notation $q^{\perp}$. Equivalently, a variety of T-algebras is a class of T-algebras orthogonal to a regular epi $q : FX \twoheadrightarrow Q$,[1] a definition which dates back to [14]. With this terminology in place we state Birkhoff's famous HSP theorem.

▶ **Theorem 1** ([17, 87, 49, 6, 7]). *Let $T : \mathsf{Set} \to \mathsf{Set}$ be a varietor. A class of T-algebras is a variety iff it is closed under Homomorphic images (H), Subalgebras (S), and Products (P).*

▶ **Example 2.** Recall that a monoid is a set M equipped with a binary operation $* : M \times M \to M$ and a constant $e \in M$ satisfying the three *equations*: $x * (y * z) = (x * y) * z$, $e * x = x$, and $x * e = x$. Every monoid is an algebra for the functor $\Sigma M = M \times M + 1$, and the functor Σ is a varietor: F_Σ builds the set of all formal terms constructed from the signature and a set of variables (e.g. $\{x, y, z\}$), and equips it with the trivial Σ-algebra structure taking the unit to be the *term* e, and the product of two terms s, t to be the *term* $s * t$. The equations of the theory of monoids can be described as the pair of maps $e_1, e_2 : 3 \rightrightarrows UF\{x, y, z\}$, where $3 \triangleq \{0, 1, 2\}$ and for $0 \leqslant i \leqslant 2$, $e_1(i)$ (resp. $e_2(i)$) picks the left-hand-side (resp. right-hand-side) of the i^{th} equation above. A monoid is a Σ-algebra in the variety defined by the coequalizer of the adjoint morphisms $\hat{e}_1, \hat{e}_2 : F3 \rightrightarrows F\{x, y, z\}$ which homomorphically sends formal terms on 3 to terms on $\{x, y, z\}$, for example $\hat{e}_1(1 * 2) = (e * x) * (x * e)$. The relation defined by the span $U\hat{e}_1, U\hat{e}_2 : UF3 \rightrightarrows UF\{x, y, z\}$ is thus closed under the rule

$$\frac{s_1 = t_1 \quad s_2 = t_2}{s_1 * s_2 = t_1 * t_2}\ \textbf{p-cong}$$

For example, $(x * e) * (e * x) = x * x$ is an equation belonging to this relation. Such a relation on terms is called a *pre-congruence* in [49].

The rule **p-cong** generalizes easily to all polynomial functors, but it is not obvious how it should be adapted to the general case. Therefore, we simply say that the relation defined by a span on UFX is a *pre-congruence* if it is of the shape $U\hat{e}_1, U\hat{e}_2 : UFE \rightrightarrows UFX$.

It is important to note that the quotient (Q, ν) in (1) is *not* a member of the variety. In Example 2, $y * e$ and y belong to different equivalence classes in Q since the quotient q only needs to identify $x * e$ and x. The interpretations of $y * e$ and y are equal in all objects belonging to the variety of monoids because of the universal quantification over the morphism f in (1) which takes care of all substitutions. In order to build a quotient that *does* belong to the variety we need more equations than those in the set UFE. It is well known that

[1] By taking $U \ker(q) \rightrightarrows UFX$ as the set of equation and using the fact that U is monadic [5, 20.56], it can be shown that we recover the quotient q as the coequalizer of the lifted equations.

equational reasoning also adheres to the following rules:

$$\frac{}{t = t}\ \textbf{ref} \qquad \frac{t = s}{s = t}\ \textbf{sym} \qquad \frac{s = t \quad t = u}{s = u}\ \textbf{trans}$$

A relation on UFX which is closed under **ref**, **sym**, and **trans** is called an *equivalence relation*, and a pre-congruence which is also an equivalence relation is called a *congruence*. It is easy to turn the relation defined by (1) into a congruence by taking the *kernel pair* of the coequalizer q, i.e. by moving to the exact sequence $\ker(q) \rightrightarrows FX \twoheadrightarrow (Q, \nu)$. Since any coequalizer is also the coequalizer of its kernel pair, q remains the coequalizer, and thus the variety it defines remains the same. We have now increased the collection of derivable equations. Following Example 2, the congruence $\ker(q) \rightrightarrows F\{x, y, z\}$ contains the equation $e * x = x * e$, for example, which requires **sym** and **trans** to derive.

As the reader will have guessed, we need to add substitution instances. Starting from the pre-congruence of (1), this can be done categorically [25, §1.4] by considering the coequalizer

$$\coprod_{v \in V} FE \underset{[\hat{v} \circ \hat{e}_2]_{v \in V}}{\overset{[\hat{v} \circ \hat{e}_1]_{v \in V}}{\rightrightarrows}} FX \xrightarrow{\quad q' \quad} (Q', \nu') \tag{2}$$

where V is the set of all substitutions $V = \{v : X \to UFX\}$. It is not difficult to see that (Q', ν') now *does* belong to the variety defined by $q' : FX \twoheadrightarrow (Q', \nu')$, i.e. $q' \perp (Q', \nu')$.

Since F is a left-adjoint, $\coprod_{v \in V} FE \simeq F(\coprod_{v \in V} E)$, i.e. (2) involves the free T-algebra generated by all substitution instances of the axioms UFE. The relation defined by the span $U \coprod_{v \in V} FE \rightrightarrows UFX$ is a pre-congruence closed under the substitution rule

$$\frac{s = t \qquad v \in V}{\hat{v}(s) = \hat{v}(t)}\ \textbf{subst}$$

Applying (2) to Example 2, we get that $y * e = y$ now belongs to the stock of equations. In fact, it appears several times, since any substitution mapping x to y will produce it.

Following [49] we say that a set of equations $e_1, e_2 : E \rightrightarrows UFX$ is *stable* if it is closed under substitutions in the sense that for any $v \in V$ there exists a (necessarily unique) map $\tilde{v} : E \to E$ such that $e_i \circ \tilde{v} = U\hat{v} \circ e_i, i = 1, 2$. Taking the kernel pair

$$\ker(q') \rightrightarrows FX \xrightarrow{\quad q' \quad} (Q', \nu') \tag{3}$$

constructs a stable set of equations $U \ker(q') \rightrightarrows UFX$ by construction [25, §1.4].

We have described three categorical constructions – lifting the equations, closing under (2), and taking the kernel pair of the coequalizer (3) – which, combined, turn a *set* of equations $e_1, e_2 : E \rightrightarrows UFX$ into an exact sequence $\ker(q') \rightrightarrows FX \twoheadrightarrow (Q', \nu')$ which defines a *stable congruence* $U \ker q' \rightrightarrows UFX$. We do not know if this purely categorical procedure produces the *smallest* stable congruence, and we do not know if the order in which the three steps are carried out matters. These questions are also raised in [48, §8], and as far as we could see, no simple categorical argument can answer them. What is clear however, is that this construction defines a quotient of the free T-algebra which belongs to the variety it defines.

▶ **Theorem 3** ([49] Thm 3.5.3). *Let T be a varietor, let $e_1, e_2 : E \rightrightarrows UFX$ be a stable set of T-equations over X, and consider the coequalizer*

$$FE \underset{\hat{e}_2}{\overset{\hat{e}_1}{\rightrightarrows}} FX \xrightarrow{\quad q \quad} (Q, \nu).$$

Then $q \perp (Q, \nu)$. Conversely, if $q \perp (Q, \nu)$, then $\ker(q)$ is stable.

We finish by stating Birkhoff's completeness theorem for equational reasoning. Given a collection $\mathbb{V}$ of T-algebras (e.g. a variety), define $\mathrm{Eq}(\mathbb{V})$ as the set of equations satisfied by every T-algebra in $\mathbb{V}$, i.e.

$$\mathrm{Eq}(\mathbb{V}) = \{e_1, e_2 : 1 \rightrightarrows UFX \mid \forall (M,\alpha) \in \mathbb{V}, \forall v : X \to M, U\hat{v} \circ e_1 = U\hat{v} \circ e_2\}.$$

▶ **Theorem 4** (Birkhoff's completeness theorem, e.g. [17, 87, 49])**.** *Let Σ be a polynomial functor, and let $E \rightrightarrows U_\Sigma F_\Sigma X$ be a set of equations. Then $E = \mathrm{Eq}(\mathbb{V})$ for some variety $\mathbb{V}$ iff E is closed under **p-cong**, **subst**, **ref**, **sym** and **trans** .*

3.2 Coequations, corelations and covarieties of coalgebras

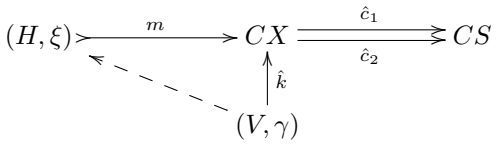
Next, we dualize the concepts developed in §3.1. We denote by $\mathsf{CoAlg}(T)$ the category of T-coalgebras and T-coalgebra homomorphisms. A functor $T : \mathsf{Set} \to \mathsf{Set}$ is called a *covarietor* [6] if the forgetful functor $U_T : \mathsf{CoAlg}(T) \to \mathsf{Set}$ has a *right* adjoint $C_T : \mathsf{Set} \to \mathsf{CoAlg}(T)$, called the *cofree functor*. For any set X, the coalgebra C_TX is called the *cofree coalgebra over the set of colours X*, or the *cofree coalgebra in X colours*. We omit subscripts if there is no risk of confusion. Intuitively, C_TX is the collection of *X-patterns*, T-processes (histories of states in a T-transition system) whose states are labelled by elements of X.

Given a covarietor T we dualize the notion of equation by defining a *T-coequation in X colours* [95] to be a *cospan* $c_1, c_2 : UCX \rightrightarrows 2$. The role of $2 \triangleq \{0,1\}$ is dual to the role of 1 in the definition of a T-equation, since 2 is a cogenerator in Set, whilst 1 is a generator. Following [95], we define a *T-coequational specification* S as a pair of maps $c_1, c_2 : UCX \rightrightarrows S$. This concept dualizes the notion of a set of T-equations, and whilst a set of T-equations is equivalent to a relation on UFX, a coequational specification defines a *corelation*, i.e. a map $UCX + UCX \twoheadrightarrow S$. One should think of a corelation on UCX as two different classification schemes – in the case of a coequation, two binary classification schemes, accepting or rejecting behaviours – used to select the behaviours/patterns that they cannot distinguish.

The dual to the notions of valuation and interpretation/substitution are the notions of *colouring* and *recolouring map*. Given a T-coalgebra (V,γ), a function $k : V \to Y$ is called a *Y-colouring map*, as it labels the states of the coalgebra with the colours of Y. Given such a colouring map, we call its cofree extension $\hat{k} : (V,\gamma) \to CY$ a *recolouring map*, $\hat{k} \triangleq C_Tk \circ \eta^T_{(V,\gamma)}$. Starting at a state $v \in V$, this map follows the history of the T-transition system (V,γ), reads the colour(s) of the successor state(s) at each time step, and uses this information to construct the T-transition system of observed colours. The original T-history is typically infinite, and therefore so is the T-history of its colours. Thus, $\hat{k}$ is a map which typically processes an entire infinitary structure in one go.

Colouring maps of the shape $k : UCX \to Y$ are important to understanding coequations. Imagine an omniscient being that can examine the entire (possibly infinite) history of an X-pattern in UCX and then classify it according to a rule of her choosing with a set of labels Y. This is a colouring map on UCX. In particular, for a colouring map $k : UCX \to X$, the omniscient being can examine the entire X-labelling history of a T-process and aggregate this information into a single X-label. Every cofree coalgebra comes with such a canonical colouring map $\varepsilon^T_X : UCX \to X$ provided by the counit ε^T of the adjunction $U \dashv C$, which returns the colour of the initial state. Given a colouring map $k : UCX \to Y$, the associated recolouring map $\hat{k} : CX \to CY$ can be understood as the process by which our omniscient being can follow an entire T-process and, at each time-step, classify the *remaining history* of the T-process according to the colouring map k. In this way the omniscient being can build the *labelling history* of the entire process in one single evaluation.

The *covariety defined by a T-coequational specification $c_1, c_2 : UCX \rightrightarrows S$ over a set (of colours) X*, is the class of coalgebras (V, γ) such that for every colouring map $k : V \to X$, there is a unique coalgebra morphism from (V, γ) to the *equaliser* (H, ξ) of $\hat{c}_1, \hat{c}_2$ such that



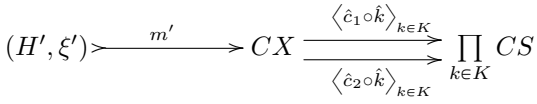
(4)

commutes.[2] The coalgebra (V, γ) is said to be co-orthogonal to the regular mono m, written $m \top (V, \gamma)$, and the covariety defined by (4) can be described as the collection of coalgebras $m^\top$ which are co-orthogonal to m. With this definition, we can state the dual to Theorem 1:

▶ **Theorem 5** (co-Birkhoff (HSC) theorem, e.g. [65, 49, 6, 7]). *Let T : Set → Set be a covarietor. A class of T-coalgebras is a covariety iff it is closed under Homomorphic images (H), Subcoalgebras (S) and Coproducts (C).*

Dual to the case of varieties, closure properties of the corelation $c_1, c_2 : UCX \rightrightarrows S$ can ensure that we obtain an equaliser which belongs to the covariety. Most of the literature focuses on closure properties on the subobject side of (4), e.g. the notion of mongruence [50], the (modal) closure operators of [49, 48], and the notion of invariant subcoalgebra of [45]. Since we dedicate §4 to this perspective, we follow [95, 65] and focus on the quotient.

Merging the nomenclatures of [49] and [65], we call the cospan $\hat{c}_1, \hat{c}_2 : CX \rightrightarrows CS$ a *pre-cocongruence*, the notion dual to a pre-congruence. Thus a pre-cocongruence is a corelation that has a coalgebra structure compatible with that of CX, i.e. which is closed under taking successors.[3] Following [95], we will say that a corelation $c_1, c_2 : UCX \rightrightarrows S$ is *coreflexive* if there exists a map $s : S \to UCX$ such that $s \circ [c_1, c_2] = [\mathrm{id}_{UCX}, \mathrm{id}_{UCX}]$, i.e. if it is only allowed to identify a behaviour $(1, t)$ in the first component of the coproduct with a behaviour $(2, s)$ in the second if $t = s$. There is also a notion of cosymmetric, cotransitive, and of coequivalence corelation, but it turns out that in Set these are implied by being coreflexive [95]. Following our earlier definition of pre-congruence and congruence, we will say that a pre-cocongruence is a cocongruence if it is coreflexive. It is easy to turn any corelation into a coreflexive corelation, it suffices to consider the *cokernel of its equaliser*. Finally, dual to the notion of a stable set of equations, we will say that a corelation $c_1, c_2 : UCX \rightrightarrows S$ is *invariant* if for any colouring map $k : UCX \to X$ there exists a (necessarily unique) morphism $\tilde{k} : S \to S$ such that $c_i \circ \tilde{k} = U\hat{k} \circ c_i, i = 1, 2$. By dualizing (2)-(3) we can turn any corelation into an invariant corelation by first considering the equalizer of the cospan



(5)

where $K = \{k : UCX \to X\}$ is the set of X-colouring maps. Since C is right-adjoint, it preserves products and $\prod_k CS \simeq C \prod_k S$. Thus, we are considering as corelation a pair of maps which can perform two "S-classifications" of an X-pattern *and all its X-recolourings*, simultaneously. Clearly, $m' \top (H', \xi')$, so (H', ξ') belongs to the covariety it defines.

[2] For any Set-endofunctor the category CoAlg(T) always has equalisers [43, 5.1]
[3] This is what Kurz calls a cocongruence in [65].

By taking the cokernel pair of the equalizer m' above $(H', \xi') \rightarrowtail CX \rightrightarrows \mathrm{coker}(m')$ we get a corelation which is invariant by construction. In fact, we get an *invariant cocongruence*, which are to cofree coalgebras what stable congruences are to free algebras. We can now state the dual of Theorem 3:

▶ **Theorem 6.** *Let T be a covarietor, let $c_1, c_2 : UCX \rightrightarrows S$ be an invariant T-coequational specification, and consider the equalizer*

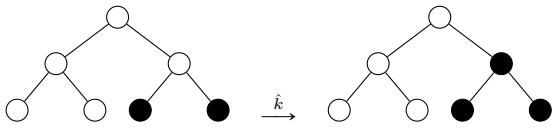
$$(H, \xi) \rightarrowtail^{m} CX \underset{\hat{c}_1}{\overset{\hat{c}_2}{\rightrightarrows}} CS.$$

Then $m \top (H, \xi)$. Conversely, if $m \top (H, \xi)$, then $\mathrm{coker}(m)$ *is invariant.*

▶ **Example 7.** Let us consider the same endofunctor as in Example 2, namely the functor $\Sigma X = X \times X + 1$. This functor is a covarietor and the cofree Σ-coalgebra $C_\Sigma X$ over X is the set of finite and infinite binary trees whose nodes are labelled by the elements of X [7]. Consider in particular the cofree Σ-coalgebra over a set of two colours, which we shall write as $\{b, w\}$ for "black" and "white". We now define the coequation $c_1, c_2 : UC\{b, w\} \rightrightarrows 2$

$$c_1(t) = \begin{cases} 1 & \text{if } \mathsf{LeftChild}(t) \text{ is labelled } b \\ 0 & \text{else} \end{cases} \qquad c_2(t) = \begin{cases} 1 & \text{if } \mathsf{RightChild}(t) \text{ is labelled } b \\ 0 & \text{else} \end{cases}$$

where $\mathsf{LeftChild}(t)$ being labelled b assumes that it exists, i.e. that t is not a leaf state, and similarly for $\mathsf{RightChild}(t)$. Recall that a *coequation-as-corelation* defines the set of behaviours which cannot be distinguished by the two classification schemes. The coequation above defines the covariety of finite and infinite binary trees with the property that if a state has left and a right children states, then they must be equal; i.e. the covariety of deterministic binary trees. To see this, consider the equalizer $m : (H, \xi) \to CX$ of $\hat{c}_1, \hat{c}_2$. It contains all binary trees such that left- and right-successors share a colour. Its cokernel defines the cocongruence on $CX + CX \to C\{b, w\}$, which only identifies $(1, s)$ and $(2, t)$ if $s = t$ belongs to H. This cocongruence is not invariant: the two copies in $CX + CX$ of the left-hand tree t_l below are identified by the corelation c_1, c_2 (and its coreflexive closure), but can be recoloured into two copies of the right-hand tree t_r, which will be kept distinct by the corelation.



By constructing the invariant closure of the corelation using the construction of (5), the two trees above become components in the tuple of all recolourings of t_l. Applying $\hat{c}_1$ and $\hat{c}_2$ component-wise to this tuple will yield two tuples which will disagree at the coordinate of t_r. In fact, the equalizer (H', ξ') of $\langle \hat{c}_1 \circ \hat{k} \rangle_{k \in K}, \langle \hat{c}_2 \circ \hat{k} \rangle_{k \in K} : C\{b, w\} \rightrightarrows \prod_k C2$ contains precisely the trees whose nodes at depth n all have the same colour, in other words the equalizer is isomorphic to $\{b, w\}^* \cup \{b, w\}^\omega$. From this it follows that given an arbitrary Σ-coalgebra (V, γ), if *any* recolouring map $\hat{k} : (V, \gamma) \to C\{b, w\}$ has to factor through $\{b, w\}^* \cup \{b, w\}^\omega$ it must be the case that $\pi_1(\gamma(x)) = \pi_2(\gamma(x))$ or $\gamma(x) \in 1$ for all $x \in V$.

4 Coequations-as-predicates

The coequations-as-predicates paradigm provides a picture of coequations that is very flexible with regard to how they can be written. In this paradigm, a coequation is a subset of a cofree coalgebra, so any method of describing subsets can be used. In practice, the elements of a cofree coalgebra carry some structure, for eg. a tree or a stream of numbers, allowing the user to describe coequations in terms of this structure.

In §4.1 and §4.2, we will see some examples of predicate coequations and their descriptions. We then give a brief account of Adámek and Schwencke's observation that there is an inherent logical structure to coequations in §4.3. Finally, in §4.4, we talk about the expressiveness of predicate coequations in general, and give a generalization of predicate coequations when there are no cofree coalgebras.

In this section, we entirely focus on coalgebras in Set. Many of the results can be generalized to coalgebras over other base categories, see for example [7] and the thesis [49].

4.1 Behavioural coequations

Fix a covarietor T on Set with forgetful-cofree adjunction $U \dashv C$, and let (Z, δ) denote the final coalgebra $C1$. Given two coalgebras $(V_1, \gamma_1), (V_2, \gamma_2)$, two states $v_1 \in V_1$ and $v_2 \in V_2$ are said to be *behaviourally equivalent* (e.g. [62]) if there is a third coalgebra (V', γ') and homomorphisms $h_i : V_i \to V'$ such that $h_1(v_1) = h_2(v_2)$. Behavioural equivalence is an equivalence relation: it is reflexive and symmetric, and since the category $\mathsf{CoAlg}(T)$ has pushouts, it is also transitive. Furthermore, since every coalgebra (V, γ) admits a unique coalgebra homomorphism $!_V : (V, \gamma) \to (Z, \delta)$, the state $!_V(v)$ of Z is a representative of the behavioural equivalence class of v for any state $v \in V$. This is the motivation for calling the states of (Z, δ) *behaviours* (for the functor T).

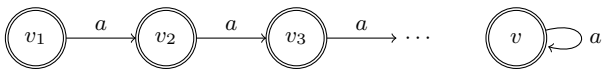
A *behavioural coequation* is a subset of Z. A coalgebra (V, γ) *satisfies* $W \subseteq Z$, written $(V, \gamma) \models W$, if $\mathrm{im}(!_V) \subseteq W$, i.e. a coalgebra satisfies a behavioural coequation if the coequation contains all of the behaviours exhibited by the coalgebra. A *behavioural covariety* is a class of coalgebras of the form $\mathsf{Cov}(W) = \{(V, \gamma) \mid (V, \gamma) \models W\}$ for some $W \subseteq Z$.

▶ **Example 8.** Coalgebras for the functor $T_{det} = 2 \times \mathrm{Id}^A$ are deterministic automata. The final deterministic automaton is $(2^{A^*}, \langle \epsilon?, \partial \rangle)$, where A^* is the set of empty or nonempty words in the alphabet A (here, ϵ is the empty word), and

$$\epsilon?(L) = \begin{cases} 1 & \text{if } \epsilon \in L \\ 0 & \text{otherwise} \end{cases} \qquad \partial(L)(a) = \{w \in A^* \mid aw \in L\}$$

for any $L \subseteq A^*$ and $a \in A$ [79, 19]. Recall that the set $\mathsf{Reg} \subseteq 2^{A^*}$ of *regular languages* is the smallest subset of 2^{A^*} closed under concatenation, iteration, and finite unions, and containing $\{\epsilon\}$ and $\{a\}$ for each $a \in A$. The class $\mathsf{Cov}(\mathsf{Reg})$ of deterministic automata that accept regular languages is a behavioural covariety.

By Kleene's theorem, a deterministic automaton satisfies Reg if and only if it is bisimilar to a locally finite automaton. There are many examples of deterministic automata that satisfy Reg but are not locally finite: The following automata are bisimilar and both accept the language a^*, but only one of them is locally finite.



The relationship between regular languages and finite automata in Example 8 is typical of behavioural coequations. Behavioural coequations constrain dynamics by constraining behaviour, and behaviour is preserved under many of the useful operations on coalgebras. In general, behavioural coequations carve nicely structured categories out of $\mathsf{CoAlg}(T)$. Following the co-Birkhoff result of Theorem 5, we will say that a class of coalgebras is a *structural covariety* if it is closed under homomorphic images, subcoalgebras and coproducts. Note that this concept makes sense whether T is a covarietor or not.

▶ **Proposition 9** (Rutten [79]). *For any $W \subseteq Z$, $\mathsf{Cov}(W)$ is a* structural covariety.

However, not every structural covariety is carved out of $\mathsf{CoAlg}(T)$ by a behavioural coequation. As we saw in Example 8, locally finite deterministic automata are not behaviourally specified: if they were, their defining coequation would be Reg. On the other hand, we will see in § 4.2 that (under mild conditions) locally finite coalgebras form a structural covariety.

The mismatch between behavioural coequations and covarieties does not detract from the importance of behavioural constraints in the computer science literature. Behavioural coequations are particularly common in fields like automata theory and process algebra where specification languages play an important role.

▶ **Example 10.** Fix a set A of *atomic actions*, and consider the following BNF grammar

$$E ::= a \in A \mid x \in \mathrm{Var} \mid E + E \mid a(E) \mid \mu x.F \qquad F ::= a \in A \mid F + F \mid a(E) \mid \mu x.F$$

A *specification* is an expression $e \in E$ in which every variable $x \in \mathrm{Var}$ appears within the scope of a μx. The set of specifications can then be given the structure of a $\mathcal{P}_\omega(\{\checkmark\}+\mathrm{Id})^A$-coalgebra (Exp, ∂) using the GSOS law below. For any $a \in A$, $e, e_1, e_2, f \in \mathrm{Exp}$, infer

$$\frac{}{a \xrightarrow{a} \checkmark} \qquad \frac{}{a(e) \xrightarrow{a} e} \qquad \frac{e_1 + e_2 \xrightarrow{a} f}{e_i \xrightarrow{a} f} \qquad \frac{\mu x.f \xrightarrow{a} e}{f[\mu x.f/x] \xrightarrow{a} e}$$

Here, $e \xrightarrow{a} \xi$ means that $\xi \in \partial(e)(a)$. The functor $\mathcal{P}_\omega(\{\checkmark\} + \mathrm{Id})^A$ has a final coalgebra (Z, δ) consisting of A-decorated trees with transitions carrying labels from A and leaves carrying the label $\checkmark$ [96]. Every specification e gives rise to a unique behaviour $!_{\mathrm{Exp}}(e)$, and therefore also a tree. In analogy with regular languages (see Example 8), one might call the set of behaviours arising from specifications the *regular* coequation. A process satisfies the regular coequation if every of its states mimics the behaviour of a specification.

Many pairs of specifications give rise to identical behaviours: For example, $e + f$ and $f + e$ are behaviourally equivalent for any e and f, and so are $f[\mu x.f/x]$ and $\mu x.f$. Studying behavioural equivalences like these is a popular topic in process algebra [35].

The reader familliar with process algebra should note that in many cases, including Example 10, behavioural equivalence and *bisimilarity* coincide. For a general T, a bisimulation between T-coalgebras (V_1, γ_1) and (V_2, γ_2) consists of a coalgebra (R, ρ) and a pair of coalgebra homomorphisms $\pi_i : (R, \rho) \to (V_i, \gamma_i)$. Image factorisations exist in Set, so every bisimulation is equivalent to one in which $R \subseteq V_1 \times V_2$ and the homomorphisms π_1, π_2 are the projections of R onto the first and second components of R [79]. If two states $v_1 \in V_1, v_2 \in V_2$ are related by a bisimulation, we say that v_1 and v_2 are *bisimilar* and write $v_1 \leftrightarrows v_2$. Important examples of bisimulations include graphs of homomorphisms: in fact, a function $V_1 \to V_2$ is a coalgebra homomorphism if and only if its graph is a bisimulation [79, 43].

▶ **Lemma 11** (Rutten [79]). *Let (V_1, γ_1) and (V_2, γ_2) be a coalgebras, and $v_1 \in V_1$ and $v_2 \in V_2$ be states. If T preserves* weak pullbacks*, then $v_1 \leftrightarrows v_2$ if and only if v_1 and v_2 are behaviourally equivalent.*

Many of the covarietors familliar to computer scientists preserve weak pullbacks, including every polynomial functor, the covariant powerset functor $\mathcal{P}$ and its κ-accessible variants $\mathcal{P}_\kappa$, and every product, coproduct, and composition of these functors [41]. Among the resulting class of functors are the deterministic automaton functor $B \times \mathrm{Id}^A$ and the nondeterministic automaton functor $B \times \mathcal{P}(\mathrm{Id})^A$ for any output set B. The added assumption that T preserve weak pullbacks leads to a rich coalgebraic theory, and much of [79] depends on it.

Under the additional assumption that T preserves weak pullbacks, Gumm and Schröder obtain the following characterisation of behavioural covarieties.

▶ **Theorem 12** (Gumm and Schröder [45]). *Let T preserve weak pullbacks. Then a structural covariety C in $\mathsf{CoAlg}(T)$ is behavioural if and only if it is* closed under total bisimulations*, i.e. if $(V_1, \gamma_1) \in \mathsf{C}$ and there is a bisimulation (R, ρ) between (V_1, γ_1) and (V_2, γ_2) such that π_1 and π_2 are surjective, then $(V_2, \gamma_2) \in \mathsf{C}$ as well.*

▶ **Example 13.** Consider the functor $T_{alt} = (\{\checkmark\} + \mathrm{Id})^{\{a,b\}}$. In a T_{alt}-coalgebra (V, γ), write $v \xrightarrow{a} v'$ if $v' = \gamma(v)(a)$ and $v \Rightarrow a$ if $\checkmark = \gamma(v)(a)$, and similarly for b. The functor T_{alt} preserves weak pullbacks, and the final T_{alt}-coalgebra is the set of all $\{a, b\}$-decorated trees t such that every node n of t has at most one $n \xrightarrow{a} n'$ transition and at most one $n \xrightarrow{b} n'$, and all other transitions are of the form $n \Rightarrow a$ or $n \Rightarrow b$. The coalgebra structure δ is the obvious parent-child transition structure.

The class of *sequence* coalgebras, consisting all those T_{alt}-coalgebras (V, γ) such that $|\gamma^{-1}(\checkmark)| = 1$, is closed under total bisimulations. By Theorem 12, the covariety of sequence coalgebras is determined by a set W_{seq} of behaviours. One way to describe this coequation is as follows: W_{seq} is the set of all behaviours t such that the first *layer* of t is one of

Of course, many of the trees in W_{seq} are *not* behaviours exhibited by sequence T_{alt}-coalgebras, as nodes from deeper layers might accept too many or too few of a, b. This can easily be fixed once we have made the following observation.

▶ **Lemma 14** (Rutten [79]). *Let $(V, \gamma), (V', \gamma')$ be T-coalgebras and $h : V \to V'$ be a T-coalgebra homomorphism. Then $h(V)$ is a subcoalgebra of (V', γ').*

Consequently, if $(V, \gamma) \models W$ for some $W \subseteq Z$, then $!_V(V)$ is a subcoalgebra of (Z, δ) contained in W. It follows from $\mathsf{CoAlg}(T)$ being cocomplete and having image factorisations that an arbitrary union of subcoalgebras of a fixed coalgebra is a subcoalgebra. In particular, there is a largest subcoalgebra $\Box W$ contained in W, and it satisfies $\mathsf{Cov}(\Box W) = \mathsf{Cov}(W)$ for any $W \subseteq Z$. In fact, $\Box W$ is the final object of $\mathsf{Cov}(\Box W)$! The operator $\Box$ is known as the "henceforth" operator in [51], and is studied in more general settings by Hughes in [48].

4.2 Beyond behaviour

A *predicate coequation in X colours* is a set of *X-patterns*, or a subset of the cofree coalgebra CX. A coalgebra (V, γ) *satisfies* the predicate coequation $W \subseteq CX$ if, for any colouring $c : V \to X$, the homomorphism $\hat{c} : (V, \gamma) \to CX$ induced by the adjunction $U \dashv C$ factors through W (that is, $\hat{c}(V) \subseteq W$).

▶ **Theorem 15** (Rutten [79], Gumm [41]). *Let T be a covarietor, X be a set, and $W \subseteq UCX$. The class $\mathsf{Cov}(W) = \{(V,\gamma) \mid (\forall c : V \to X)\ \hat{c}(V) \subseteq W\}$ is closed under subcoalgebras, coproducts, and homomorphic images.*

Under mild assumptions on T, every structural covariety is presentable by a coequation. The key to proving the converse to Theorem 15 is to give an upper bound on the number of colours that only depends on T. This is possible when T is *κ-bounded* for some cardinal κ, meaning that for any coalgebra (V,γ) and any state $v \in V$, there is a subcoalgebra S of (V,γ) such that $v \in S$ and $|S| \leqslant \kappa$. An endofunctor on Set is *bounded* if it is κ-bounded for some κ.[4] The class of bounded functors is broad enough to capture most functors in everyday use by computer scientists [79, 42]. At the beginning we assumed that T is a covarietor, but boundedness actually implies this.

▶ **Theorem 16** (Kawahara & Mori [57]). *If $T : \mathsf{Set} \to \mathsf{Set}$ is bounded, then T is a covarietor.*

Let C be a structural covariety and κ an infinite cardinal, and assume that T is κ-bounded. Given a state v of a coalgebra (V,γ), there is a subcoalgebra S of (V,γ) containing v such that $|S| \leqslant \kappa$. By a simple renaming of states, S is isomorphic to a coalgebra whose state space is a set of numbers in κ. This means that every T-coalgebra is locally isomorphic to a coalgebra of the form (S,σ) where $S \subseteq \kappa$, and in particular that every coalgebra in C is locally of this form. Writing $\mathsf{G} = \{(S,\sigma) \in \mathsf{C} \mid S \subseteq \kappa\}$, the coequation $W_{\mathsf{G}} = \bigcup\{\hat{c}(S) \mid (S,\sigma) \in \mathsf{G} \text{ and } c : S \to \kappa\}$ determines C, meaning that $\mathsf{C} = \mathsf{Cov}(W_{\mathsf{G}})$.

▶ **Theorem 17** (Rutten [79]). *If T is κ-bounded and C is a structural covariety in $\mathsf{CoAlg}(T)$, then $\mathsf{C} = \mathsf{Cov}(W)$ for some $W \subseteq C\kappa$.*

Behavioural coequations are instances of predicate coequations: They are the coequations in 1 colour. We have already seen in Examples 8, 10, and 13 that some covarieties are presentable by behavioural coequations despite the type functor failing to be 1-bounded.[5] The bound on the number of colours provided by Theorem 17 is rarely optimal in practice.

Towards a better bound, recall the definition of behavioural equivalence from §4.1. Let C be a covariety that is *closed under behavioural equivalence*, meaning that it contains every coalgebra (V,γ) such that every state $v \in V$ is behaviourally equivalent to a state of a coalgebra in C. Then C is necessarily behavioural. Indeed, if $W_{\mathsf{C}} = \{!_S(x) \mid x \in S, (S,\sigma) \in \mathsf{C}\}$, then $(S,\sigma) \models W_{\mathsf{C}}$ for any $(S,\sigma) \in \mathsf{C}$. Conversely, every state of a coalgebra satisfying W_{C} behaves like a state from a coalgebra in C, and by assumption such a coalgebra must be in C.

This argument generalizes to the λ-pattern situation by saying that a class C of coalgebras is *closed under λ-pattern equivalence* if $(V,\gamma) \in \mathsf{C}$ whenever the following condition is met: for any $v \in V$ and any colouring $c : V \to \lambda$, there is a $(S,\sigma) \in \mathsf{C}$ with a state $x \in S$ and a colouring $c' : S \to \lambda$ such that $\hat{c}(v) = \hat{c}'(x)$. We thus have:

▶ **Proposition 18.** *Let C be a covariety closed under λ-pattern equivalence. Then $\mathsf{C} = \mathsf{Cov}(W)$ for some $W \subseteq UC\lambda$.*

▶ **Example 19.** Consider the deterministic automaton endofunctor $T_{det} = 2 \times \mathrm{Id}^A$ from Example 8. Fix two words $w_1, w_2 \in A^*$ and let C be the class of deterministic automata (V,γ) in which $v \xrightarrow{w_1} v'$ and $v \xrightarrow{w_2} v''$ implies $v' = v''$. Consider an automaton $(V, \langle o, \partial\rangle)$, $v \in V$, and $c : V \to 2$ the colouring $c(x) = 1 \iff x = \partial(v)(w_1)$,[6] and assume there is an automaton

[4] Equivalently, T is *accessible* or *small* [6].
[5] It is straightforward to check that each is ω-bounded, on the other hand.
[6] Here, $\partial(v) : A^* \to V$ is defined by $\partial(v)(\epsilon) = v$ and $\partial(v)(wa) = \partial(\partial(v)(w))(a)$.

$(V', \langle o', \partial' \rangle) \in \mathsf{C}$ with $v' \in V'$ and a colouring $c' : V' \to 2$ such that $\hat{c}(v) = \hat{c}'(v')$. Then $c(\partial(v)(w_2)) = c'(\partial'(v')(w_2)) = c'(\partial'(v')(w_1)) = c(\partial(v)(w_1))$, so that $\partial(v)(w_2) = \partial(v)(w_1)$ by construction of c. It follows that C is closed under 2-pattern equivalence, and is therefore determined by a coequation in 2 colours. Indeed, where ε is the counit, it is given by

$$W = \{t \in UC2 \mid \varepsilon_2(\partial(t)(w_1)) = \varepsilon_2(\partial(t)(w_2))\}.$$

In §4.1, we saw that bisimilarity and behavioural equivalence coincide when T preserves weak pullbacks. This gave way to Theorem 12, which characterised behavioural covarieties in terms of total bisimilarity. Adámek applies the same reasoning in [4] to give a bound like the one in Prop. 18 in terms of bisimulations: For a cardinal λ, say that a covariety C is *closed under λ-colour bisimilarity* if $(V, \gamma) \in \mathsf{C}$ whenever the following condition is met: For any $c : V \to \lambda$ there is a $(S, \sigma) \in \mathsf{C}$, a colouring $c' : V' \to \lambda$, and a total bisimulation (R, ρ) between (V, γ) and (V', γ') such that $c \circ \pi_1 = c' \circ \pi_2$.

▶ **Theorem 20** (Adámek [4]). *Suppose T is a covarietor that preserves weak pullbacks. A covariety in $\mathsf{CoAlg}(T)$ is presentable by a predicate coequation in λ colours if and only if it is closed under λ-colour bisimilarity.*

▶ **Example 21.** The class C of *simple* graphs (of finite degree) can be seen as a covariety in $\mathsf{CoAlg}(\mathcal{P}_\omega)$. In a $\mathcal{P}_\omega$-coalgebra (V, γ), write $v_1 \to v_2$ to denote $v_2 \in \gamma(v_1)$. Then (V, γ) is a *simple graph* if $\to$ is a reflexive symmetric relation on V.[7] Reflexivity and symmetry are given by the coequations $W_{ref} = \{t \mid (\exists s)\ \varepsilon_2(s) = \varepsilon_2(t) \text{ and } t \to s\}$ and $W_{sym} = \{t \mid (\forall s)\ (t \to s \implies (\exists u)\ \varepsilon_2(u) = \varepsilon_2(t) \text{ and } s \to u)\}$ respectively. The modal logician will recognise the axioms (T) $p \to \Diamond p$ and (B) $p \to \Box\Diamond p$, the roles of the propositional variable p being played by the colouring map ε_2. The coequation we are looking for is therefore $W_{sim} = W_{ref} \cap W_{sym}$. By Theorem 20, C is closed under 2-colour bisimilarity. This covariety is also not behavioural: $\bullet \circlearrowleft$ and $\bullet \to \bullet \to \bullet \to \cdots$ are bisimilar, for example.

▶ **Example 22.** Recall that a coalgebra (V, γ) is *locally finite* if for any $v \in V$, there is a subcoalgebra S of (V, γ) such that $v \in S$ and S is finite. If T preserves weak pullbacks, then the class C_ω of locally finite T-coalgebras is a covariety in ω colours.

To see why, let (V, γ) be a coalgebra such that for any $c : V \to \omega$, there is a locally finite (V', γ'), a colouring $c' : V' \to \omega$, and a total bisimulation (R, σ) between (V, γ) and (V', γ') such that $c \circ \pi_1 = c' \circ \pi_2$. If $v \in V$ and $c : V \to \omega$ is any colouring, and we take $(V', \gamma'), (R, \rho), c : V' \to \omega$ as before, then vRv' for some $v' \in V'$. Since (V', γ') is locally finite, $c'(S')$ is finite for some subcoalgebra S' of (V', γ') containing v'. The projection π_2 is surjective, so $P = \pi_2^{-1}(S')$ is a subcoalgebra of (R, ρ) containing (v, v'). Taking images, we see that $c(\pi_1(P)) = c'(\pi_2(P)) = c'(S')$, so that $\pi_1(P)$ is a subcoalgebra of (V, γ) with finite image under c. Since T preserves weak pullbacks, there is a smallest subcoalgebra $\langle v \rangle$ of (V, γ) containing v [79]. This subcoalgebra is contained in every P as constructed above, so has finite image under c for any c. A set is finite if and only if every image of the set under a map into ω has a finite image, so $\langle v \rangle$ must be finite. It follows that (V, γ) is locally finite, so by Theorem 20, C_ω is presentable with a coequation in ω colours. The desired coequation consists of those ω-patterns in which only finitely many colours appear.

[7] A simple graph in this sense contains the same information as the more traditional concept from combinatorics. However, $\mathcal{P}_\omega$-coalgebra homomorphisms are not graph homomorphisms. For a coalgebraic depiction of traditional directed graphs, see pg. 22 of [80], or [55].

4.3 Logic and Avoiding Patterns

As we have already seen, it is possible for distinct coequations, like $\Box W$ and W in Example 13 for instance, to specify the same covariety. In this short section, we describe Adámek and Schwencke's framing of this equivalence between coequations as a logical equivalence in [4, 89, 90], and discuss Adamék's sound and complete deduction system for the resulting logic of coequations for a polynomial endofunctor T.

Given two coequations W_1 and W_2, W_1 is said to *imply* W_2, written $W_1 \models W_2$, if for any coalgebra (V,γ), $(V,\gamma) \models W_2$ whenever $(V,\gamma) \models W_1$. For example, $W_1 \subseteq W_2$ implies $W_1 \models W_2$, and $W \models \Box W$ and $\Box W \models W$. The inference relation $\models$ also interacts with recolourings: every $h : X \to X$ induces $\hat{c} : CX \to CX$ such that $W \models \hat{h}(W)$.

Further analysis of the inference relation $\models$ is possible with a notation used by Gumm. In [41], Gumm gives a negative description of coequations, as predicates of the form $\boxminus t = (CX) - \{t\}$ for a pattern t. This is a particularly useful notation for coequations when patterns are easily described but general predicates are not. Such is the case when T is a polynomial functor, as patterns are identifiable with certain trees.

Fix a polynomial functor $T_\Sigma = \bigcup_{p\in\Sigma} \mathrm{Id}^{ar(p)}$, where κ is a cardinal, Σ is a set, and $ar : \Sigma \to \kappa$. For a set of colours X, an X-pattern is a tree t in which every node n is labelled with a pair $(p,x) \in \Sigma \times X$ and a transition function that maps each $\alpha < ar(p)$ to each child of n. The structure map of CX is given by parent→child transitions: if n is a node of t and n' is the αth child of n, then $t \xrightarrow{\alpha} s$ when s is the subtree of t rooted at n'. There is a unique node of t with no incoming transitions, called its *root*. Each node of t is the root a tree, and we call trees of this form *subtrees* of t. We write $s \sqsubseteq t$ to denote that s is a subtree of t.

Given $t, s \in UCX$, if $s \sqsubseteq t$ and $(V,\gamma) \models \boxminus s$, then $(V,\gamma) \models \boxminus t$ as well. This is because, if $\hat{c}(v) = t$ for some colouring $c : V \to X$ and $v \in V$, then every path $t \xrightarrow{\alpha_1} t_1 \to \cdots \to t_{n-1} \xrightarrow{\alpha_n} s$ is witnessed in (V,γ) by a path $v \xrightarrow{\alpha_1} v_1 \to \cdots \to v_{n-1} \xrightarrow{\alpha_n} u$ such that $\hat{c}(u) = s$.[8] Furthermore, if s is a *recolouring* of t, i.e. $s = \hat{k}(t)$ for some colouring $k : UCX \to X$, then $(V,\gamma) \models \boxminus s$ implies $(V,\gamma) \models \boxminus t$ as well. This is due to the composition $c' = k \circ \hat{c}$, where $c : V \to X$ is any colouring of (V,γ), since $s = \hat{c'}(v)$ when $t = \hat{c}(v)$. We obtain the following proof rules.

$$\frac{\vdash \boxminus s \qquad t \to s}{\vdash \boxminus t}\ \textbf{child} \qquad \frac{\vdash \boxminus s \qquad s = \hat{k}(t)}{\vdash \boxminus t}\ k\textbf{-rec}$$

For any $t, s \in UCX$, if $\vdash \boxminus t$ can be deduced from $\vdash \boxminus s$ with k-**rec** (here, k is allowed to vary) and **child**, we write $\boxminus s \vdash \boxminus t$.

▶ **Theorem 23** (Adámek [4]). *For a polynomial endofunctor T_Σ, a set X, and any $s, t \in CX$, $\boxminus s \models \boxminus t$ if and only if $\boxminus s \vdash \boxminus t$.*

As shown in Adámek's [4] and Schwencke's [89, 90], the logic of coequations for polynomial functors (described above) can be extended to include many bounded functors. The extended logic relies on the fact that every bounded functor is a natural quotient of some polynomial functor [8], and by extension every cofree coalgebra for a bounded functor is a quotient of a cofree coalgebra for a polynomial functor. The subtree and recolouring rules apply to representatives, and with the right natural quotient[9] $T_\Sigma \Rightarrow T$, the ensuing logic is sound and complete with respect to coequational reasoning.

[8] Here, $v \xrightarrow{\alpha} u$ in (V,γ) if $u = \gamma(v)(\alpha)$.
[9] Namely, a so-called *regular presentation*. See [90] for details.

4.4 Generalized coequations

If T is not bounded, T is likely not a covarietor. In such a case, we cannot always use predicates to specify classes of coalgebras over the base category Set. However, as Aczel and Mendler showed in [2], every endofunctor on Set extends to a covarietor on the category of classes. By *approximating* cofree coalgebras, which may be proper classes in the case that T is unbounded, Adámek recovers *generalized coequations* in [3], and shows they are sufficient for specifying structural covarieties in general.

To approximate the cofree coalgebra in X colours, we follow Barr in [16] and construct its *final sequence*, the ordinal-indexed diagram

$$X_0 \xleftarrow{\phi_0^1} X_1 \xleftarrow{\phi_1^2} X_2 \longleftarrow \cdots \longleftarrow X_\omega \xleftarrow{\phi_\omega^{\omega+1}} X_{\omega+1} \longleftarrow \cdots$$

Here, $X_0 = 1$ and $\phi_0^1 = !$, $X_{\alpha+1} = X \times TX_\alpha$, and $\phi_{\alpha+1}^{\beta+1} = \mathrm{id}_X \times T(\phi_\alpha^\beta)$ for any ordinals $\alpha < \beta$, and $(X_\lambda, \{\phi_\alpha^\lambda\}_{\alpha<\lambda}) = \varprojlim\{\phi_\alpha^\beta : X_\beta \to X_\alpha \mid \alpha < \beta < \lambda\}$ for λ a limit ordinal.

For any T-coalgebra (V, γ) and any colouring $c : V \to X$, let $c_0 = ! : V \to X_0$ be the unique such function, and define $c_{\alpha+1} = \langle c, T(c_\alpha) \circ \gamma \rangle$ at successor ordinals, and $c_\lambda : V \to X_\lambda$ to be the unique cone homomorphism $\{c_\alpha\}_{\alpha<\lambda} \to \{\phi_\alpha^\lambda\}_{\alpha<\lambda}$ when λ is a limit ordinal. A *generalized X-pattern* is an ordinal indexed sequence $\{t_\alpha\}_{\alpha\in\mathsf{Ord}}$ such that $t_\alpha \in X_\alpha$, and $t_\alpha = \phi_\alpha^\beta(t_\beta)$ for any $\alpha < \beta$. A *generalized coequation* is a class W of generalized patterns, and $(V, \gamma) \models W$ if for any $c : V \to X$ and $v \in V$, we find $\{c_\alpha(v)\}_{\alpha\in\mathsf{Ord}} \in W$.

▶ **Theorem 24** (Adámek [3]). *For any endofunctor T, a class C of T-coalgebras is a structural covariety if and only if there is a generalized coequation W such that $\mathsf{C} = \mathsf{Cov}(W)$.*

Generalized coequations are indeed generalisations of coequations for a covarietor. If T is a covarietor, then $\phi_\lambda^{\lambda+1}$ is an isomorphism for some ordinal λ [7].[10] As $\phi_\lambda^{\lambda+1}$ is an isomorphism, it has an inverse $\langle k, \delta^X \rangle : X_\lambda \to X \times TX_\lambda$, and the cofree coalgebra CX is precisely (X_λ, δ^X). In this setting, c_λ and $\hat{c}$ coincide, and the satisfaction relation from § 4.2 coincides with the satisfaction of generalized coequations. Note, however, that while the set of colours is fixed in Theorem 17, the colours appearing in Theorem 24 can vary.

▶ **Example 25.** The powerset functor $\mathcal{P}$ is not bounded, as no $\phi_\lambda^{\lambda+1}$ can be a bijection. Nevertheless, the class of simple graphs from Example 21 forms a covariety. The presenting coequation is 2-coloured and can be visualised as a subset of X_3. Here, $X_3 = 2\times\mathcal{P}(2\times\mathcal{P}(2\times 2))$, so elements of X_3 can be thought of as extensional trees with 2-coloured nodes and height at most 3. The desired subset, call it W, is obtained by restricting the coequation in Example 21 to such trees. The generalized coequation W_{sim} then consists of all ordinal-indexed sequences $\{t_\alpha\}_{\alpha\in\mathsf{Ord}}$ such that $t_3 \in W$.

▶ **Example 26.** For a set V, let $\mathcal{F}V$ be the set of *filters* on V, upwards-closed subsets of $\mathcal{P}(V) - \{\varnothing\}$ that are closed under pairwise intersection. For a function $f : V \to V'$, let $\mathcal{F}(f)(F) = [f(F)]_{\mathrm{fil}}$ be the smallest filter containing $\{f(s) \mid s \in F\}$. Then $\mathcal{F}$ is an unbounded functor. As Gumm points out in [42], the category Top of topological spaces and open continuous maps is a covariety of $\mathcal{F}$-coalgebras. The structure map of a topological space sends every point to its filter of neighbourhoods. The coequation presenting Top appears in [68] in modal form, but in principle can be translated into a generalized coequation.

[10] Worrell shows in [96] that if T is κ^+-bounded, $\phi_\lambda^{\lambda+1}$ is an isomorphism when $\lambda = \kappa + \kappa$.

5 Coequations-as-equations

Our main sources for this section are [50, 47, 20, 78]. The last two papers are written in the language of *visible and hidden sorts*, making them relatively difficult to read. Here, we follow the single-sorted setup of [50, 47]. The generalisation to multiple sorts (i.e. to the category Set^S for some set of sorts S) presents only notational difficulties.

Following [50], let $\mathtt{At}$ be a set of atomic types and consider the grammars of types:

$$\mathtt{S} ::= \mathtt{A} \in \mathtt{At} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{S} + \mathtt{S} \mid \mathtt{S} \times \mathtt{S} \qquad \mathtt{T} ::= \mathtt{A} \in \mathtt{At} \mid \mathtt{0} \mid \mathtt{1} \mid \mathtt{T} + \mathtt{T} \mid \mathtt{T} \times \mathtt{T} \mid \mathtt{X}$$

A *destructor signature* is a set of pairs of types $\sigma_i \triangleq (\mathtt{S}_i, \mathtt{T}_i), i \in I$, called *destructors*. The interpretation of a type is determined inductively given interpretations $[\![\mathtt{A}]\!]$ of $\mathtt{A} \in \mathtt{At}$ and $[\![\mathtt{X}]\!]$ as sets, and by taking $+$ to be the coproduct, $\times$ the product, 0 the empty set, and 1 the set $\{0\}$ in Set. An interpretation of a destructor $\sigma \triangleq (\mathtt{S}, \mathtt{T})$ is a map $[\![\sigma]\!] : [\![\mathtt{S}]\!] \times [\![\mathtt{X}]\!] \to [\![\mathtt{T}]\!]$, and an interpretation of a destructor signature is an interpretation of each of its destructors.

As Set is Cartesian closed, every destructor can equally be interpreted as a map $[\![\sigma]\!] : [\![\mathtt{X}]\!] \to [\![\mathtt{T}]\!]^{[\![\mathtt{S}]\!]}$. This means that the interpretation of a destructor signature can be described as a coalgebra for the functor

$$T\,[\![\mathtt{X}]\!] \triangleq \prod_{i \in I} [\![\mathtt{T}_i]\!]^{[\![\mathtt{S}_i]\!]} \qquad (\mathtt{T} \text{ typically depends on } \mathtt{X}).$$

▶ **Example 27.** Jacobs provides the example of a simple class for a bank account where $\mathtt{At} = \{\mathtt{N}\}$, and the destructor signature is $\{(\mathtt{1}, \mathtt{N}), (\mathtt{N}, \mathtt{X})\}$. An interpretation of this destructor signature can be defined by choosing $[\![\mathtt{N}]\!] = \mathbb{N}$ and two maps $\mathrm{bal} : [\![\mathtt{X}]\!] \to \mathbb{N}$ and $\mathrm{credit} : \mathbb{N} \times [\![\mathtt{X}]\!] \to [\![\mathtt{X}]\!]$ returning the balance on the account and crediting the account by a given amount respectively. Alternatively, the interpretation of this destructor signature can be a coalgebra for $\mathbb{N} \times \mathrm{Id}^{\mathbb{N}}$.

The definition of a *term* for a destructor signature is fairly elastic (see *op.cit.*) but includes at least the following rules. First, define for each atomic type $\mathtt{A} \in \mathtt{At}$ a set $\mathrm{Var}_{\mathtt{A}}$ of variables of type $\mathtt{A}$. We also define a *unique* variable x of type $\mathtt{X}$. The following rules [50, 20] are used to build terms in context:

1. Variables are terms of the corresponding type: if $a \in \mathrm{Var}_{\mathtt{A}}$ then $\vdash a : \mathtt{A}, \vdash x : \mathtt{X}$.
2. If $\sigma = (\mathtt{S}, \mathtt{T})$ is in the destructor signature, then $s : \mathtt{S}, t : \mathtt{X} \vdash \sigma(s, t) : \mathtt{T}$.

Any constructions which might be useful, such as projections and coprojections or built-in functions, can be added to the grammar of terms. In the case of Example 27 it is useful to add the function $(-) + (-) : \mathtt{N} \times \mathtt{N} \to \mathtt{N}$ which allows the term $n : \mathtt{N}\, x : \mathtt{X} \vdash \mathrm{bal}(x) + n : \mathtt{N}$ to be constructed. A *coequation-as-equation* is defined as an equation $a_1 : \mathtt{A}_1, \ldots, a_n : \mathtt{A}_n, x : \mathtt{X} \vdash s = t : \mathtt{T}$ between two terms of the same type, in the same context. The *interpretation of terms* follows in the obvious way from the interpretation of the destructor signature (and any other build-in operations like $(-) + (-)$) and function composition.

As in the case of *coequations-as-corelations*, the purpose of these equations is *not* to identify terms via a quotient, but to *select* certain behaviours. The connection with corelations can be made explicit by observing that every term will be typed like $a_1 : \mathtt{A}_1, \ldots, a_n : \mathtt{A}_n, x : \mathtt{X} \vdash t : \mathtt{T}$ with a context containing a *unique* variable $x : \mathtt{X}$, and variables of atomic type. This means that its interpretation $[\![t]\!]$ can always be Curried, and since an interpretation of the destructor signature is a coalgebra $\gamma : X \to TX$, we can view a *coequation-as-equation* $s = t$ as a corelation:

$$X \rightrightarrows [\![\mathtt{T}]\!]^{[\![\mathtt{A}_1]\!] \times \ldots [\![\mathtt{A}_n]\!]} \quad \text{or a pre-cocongruence} \quad (X, \gamma) \rightrightarrows C_T\,[\![\mathtt{T}]\!]^{[\![\mathtt{A}_1]\!] \times \ldots [\![\mathtt{A}_n]\!]}$$

(with parallel arrows $[\![s]\!]$ and $[\![t]\!]$)

Returning to Example 27, Jacobs gives the financially sound equation $n : \mathtt{N}, x : \mathtt{X} \vdash \mathrm{bal}(\mathrm{credit}(n, x)) = n + \mathrm{bal}(x)$. By interpreting the basic destructors as a coalgebra $\gamma : X \mapsto \mathbb{N} \times X^{\mathbb{N}}$, and Currying the interpretation of the two terms in the equation, we get a corelation $X \rightrightarrows \mathbb{N}^{\mathbb{N}}$ classifying behaviours according to what the functions $\lambda n. [\![\mathrm{bal}(\mathrm{credit}(n, x))]\!]$ and $\lambda n. n + [\![\mathrm{bal}(x)]\!]$ do at state x. The coequation-as-equation *selects* the bank accounts whose behaviours cannot be distinguished by these two functions.

6 Coequations-as-modal-formulas

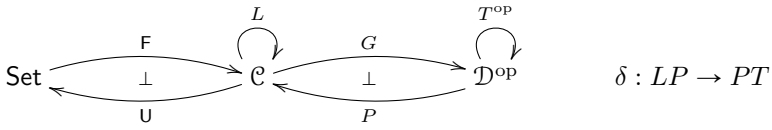
Modal logic, and its generalisation coalgebraic modal logic, is an intuitive and powerful syntax to write *predicate coequations*. We follow the abstract formalism of [60, 61, 53], as it naturally lends itself to interpreting modal formulas as coequations. In this formalism, the syntax of a modal logic is given by an endofunctor L on some base category $\mathcal{C}$, typically either the category BA of Boolean Algebras for boolean modal logics (see *op.cit.* and [74, 93, 21]), or the category DL of Distributive Lattices (see [12, 28, 27]) for positive modal logics. The base category $\mathcal{C}$ encodes all the basic logical connectives whilst the functor L constructs terms with modal operators. Since one is typically interested in finitary logics, we also make the natural assumption that L is finitary, and in particular a varietor [7, Thm. 3.17]. We thus have a free-forgetful adjunction $F_L \dashv U_L, F_L : \mathcal{C} \to \mathsf{Alg}_{\mathcal{C}}(L)$. We also assume that there exists a free-forgetful adjunction $\mathsf{F} \dashv \mathsf{U}, \mathsf{F} : \mathcal{C} \to \mathsf{Set}$, which we write in sans-serif font to keep it distinct from the other adjunctions.

▶ **Example 28.** Normal modal logic is defined by the functor $L : \mathsf{BA} \to \mathsf{BA}$ which sends a boolean algebra A to the boolean algebra of formal terms $LA = \mathsf{F}\{\Box a \mid a \in \mathsf{U}A\}/\equiv$, where $\equiv$ is the stable congruence generated by the equations $\Box\top = \top, \Box(a \wedge b) = \Box a \wedge \Box b$. Usually, A is also freely generated, specifically $A = \mathsf{FAt}$ where At is the set of propositional variables.

Coalgebraic modal logics are interpreted in coalgebras, so let us fix an endofunctor $T : \mathcal{D} \to \mathcal{D}$ describing both the kind of carrier ($\mathcal{D}$-objects) and the kind of transition systems in which modal formulas are to be interpreted. We now need two pieces of categorical data.

1. A dual adjunction $G \dashv P, G : \mathcal{C} \to \mathcal{D}^{\mathrm{op}}$ connecting the "logical category" $\mathcal{C}$ to the "model category" $\mathcal{D}$ whose objects carry the models of the interpretation.
2. A *semantic natural transformation* $\delta : LP \to PT$ to recursively compute the semantics.

The framework of coalgebraic modal logic can thus be summarized as the categorical data:



$\delta : LP \to PT$

By using P and δ, one can turn any T-coalgebra $\gamma : X \to TX$ into an L-algebra $P\gamma \circ \delta_X : LPX \to PTX \to PX$. We denote this construction, which is functorial since δ is natural, by $\widehat{P}(X, \gamma)$. The semantics can now be defined as follows: given a T-coalgebra (X, γ) and a set At of variables, a *valuation* is a map $v : \mathsf{At} \to \mathsf{U}PX$, which lifts to a $\mathcal{C}$-morphism $\hat{v} : \mathsf{FAt} \to PX$. Since PX is the carrier of $\widehat{P}(X, \gamma)$, we can re-type $\hat{v}$ as a $\mathcal{C}$-morphism $\hat{v} : \mathsf{FAt} \to U_L\widehat{P}(X, \gamma)$. Since L is a varietor, this map freely extends to a unique L-algebra morphism $[\![-]\!]_v : F_L\mathsf{FAt} \to \widehat{P}(X, \gamma)$, which recursively computes the interpretation of a modal formula in $F_L\mathsf{FAt}$ as an element of PX via the semantic transformation δ. A formula $\varphi \in F_L\mathsf{FAt}$ is said to be *satisfied at* $x \in X$ *for the valuation* v, written $(X, \gamma, x) \models_v \phi$, if $x \in [\![\phi]\!]_v$. A formula $\varphi \in F_L\mathsf{FAt}$ is said to be *valid in* (X, γ), written $(X, \gamma) \models \varphi$, if it is satisfied at every $x \in X$ and for every valuation $v : \mathsf{At} \to \mathsf{U}PX$.

▶ **Example 29.** In the case of the normal modal logic described in Example 28 we take $\mathcal{D} = \mathsf{Set}$, the dual adjunction is given by the powerset functor $\mathcal{P} : \mathsf{Set}^{\mathrm{op}} \to \mathsf{BA}$ and the ultrafilter functor $\mathcal{U} : \mathsf{BA} \to \mathsf{Set}^{\mathrm{op}}$, and models are coalgebras for the powerset functor $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$. The semantic transformation $\delta : L\mathcal{P} \to \mathcal{P}\mathcal{P}$ thus turns a modal formula over a predicate on the carrier into the set of successors which must satisfy this predicate, from the perspective of the modality. It is defined by $\delta(\Box W) = \{W' \mid W' \subseteq W\}$. Given a coalgebra $\gamma : X \to \mathcal{P}X$ and a valuation $v : \mathsf{At} \to \mathsf{U}\mathcal{P}X$, it follows from the definition of δ and $[\![-]\!]_v$ that for any $\varphi \in F_L\mathsf{FAt}$, $[\![\Box\varphi]\!]_v$ is computed recursively via $[\![\Box\varphi]\!]_v = \{x \in X : \gamma(x) \subseteq [\![\varphi]\!]_v\}$, where $[\![p]\!]_v \triangleq v(p)$. This is the usual semantics for normal modal logic, rephrased coalgebraically.

A *set* of modal axioms $\Phi \subseteq \mathsf{U}U_LF_L\mathsf{FAt}$ defines a set of equations $e_1, e_2 : \Phi \rightrightarrows \mathsf{U}U_LF_L\mathsf{FAt}$, in the sense of §3.1, since each axiom $\varphi \in \Phi$ is shorthand for the equation $\varphi = \top$, i.e. $e_1(\varphi) = \varphi, e_2(\varphi) = \top$. We can then consider the variety defined by the coequalizer $q : F_L\mathsf{FAt} \twoheadrightarrow Q$ of the free extensions $\hat{e}_1, \hat{e}_2 : F_L\mathsf{F}\Phi \rightrightarrows F_L\mathsf{FAt}$, exactly as in §3.1. We obtain, immediately from the definitions, that every set of modal axioms defines a *variety*, and these axioms are valid in a coalgebra (X, γ) precisely when $\widehat{P}(X, \gamma)$ belongs to the variety.

▶ **Proposition 30.** *Using the notation above,* $(X, \gamma) \models \Phi$ *iff* $q \perp \widehat{P}(X, \gamma)$.

A set of modal axioms Φ can also be seen as a coequation-as-predicate which defines a covariety. To see this, assume that T is a covarietor, i.e. that there exists forgetful-cofree adjunction $U_T \dashv C_T, U_T : \mathsf{CoAlg}_{\mathcal{D}}(T) \to \mathcal{D}$, and consider the cofree coalgebra $C_TG\mathsf{FAt}$ over the $\mathcal{D}$-object of colours $G\mathsf{FAt}$. The reason for choosing these colours is that by using the counit ε of the adjunction $U_T \dashv C_T$, we can construct a canonical interpretation via the adjunctions $G \dashv P$ and $F_L \dashv U_L$, and the fact that $U_L\widehat{P} \simeq PU_T$:

$$U_TC_TG\mathsf{FAt} \xrightarrow{\varepsilon} G\mathsf{FAt} \quad \Leftrightarrow \quad \mathsf{FAt} \longrightarrow PU_TC_TG\mathsf{FAt} \quad \Leftrightarrow \quad F_L\mathsf{FAt} \xrightarrow{[\![-]\!]_\varepsilon} \widehat{P}C_TG\mathsf{FAt}.$$

With this canonical interpretation map we can view Φ as a coequation-as-predicate in $G\mathsf{FAt}$ colours selecting the elements $t \in C_TG\mathsf{FAt}$ such that $(C_TG\mathsf{FAt}, t) \models_\varepsilon \varphi$ for all $\varphi \in \Phi$.

▶ **Example 31.** Returning to the classical modal logic of Examples 28 and 29, we modify the semantics slightly by considering coalgebras for an accessible version of $\mathcal{P}$, e.g. we take $T \triangleq \mathcal{P}_\kappa$ with $\kappa = |\mathsf{At}|$ so that $\mathcal{P}_\kappa\mathsf{At} = \mathcal{P}\mathsf{At}$. The coalgebraic semantics is now given in terms of a covarietor. The cofree coalgebra $C_T\mathcal{U}\mathsf{FAt} \simeq C_T\mathcal{P}\mathsf{At}$ is then the set of all κ-branching strongly-extensional trees labelled by sets of propositional variables [96]. By definition of $[\![-]\!]_\varepsilon$, if $p \in \mathsf{At}$ then $(C_T\mathcal{P}\mathsf{At}, t) \models_\varepsilon p$ iff p belongs to the set of propositional variables labelling t. The semantics of modal formulas works in the expected way, namely $\Box\varphi$ holds at a tree t if φ holds at all its children (should it have any). Thus, every modal formula $\varphi \in F_L\mathsf{FAt}$ defines the subset of $[\![\varphi]\!]_\varepsilon \subseteq U_TC_T\mathcal{P}\mathsf{At}$, i.e. a *predicate coequation*. More generally, every set Φ of modal axioms defines the predicate coequation $\bigcap_{\varphi\in\Phi} [\![\varphi]\!]_{\varepsilon^T} \subseteq U_TC_T\mathcal{P}\mathsf{At}$, containing only those trees for which all formulas in Φ are satisfied.

Now, which covarieties can be defined from a coequation-as-modal-formula in the way we just sketched? An answer to this question has long been part of the canon of modal logic, and is known as the Goldblatt-Thomason theorem [18, Thm. 3.19]. A coalgebraic version of this theorem was developed by Kurz and Rosický in [69]. This theorem can only be stated for (and is therefore only applicable to) coalgebraic modal logics such that the semantic transformation $\delta : LP \to PT$ has an inverse natural transformation $\delta^{-1} : PT \to LP$[11,12].

[11] The naturality of the inverse in not strictly necessary, but makes the presentation easier, see [69].

[12] The existence of a natural transformation $\delta^{-1} : GL \to TG$ is also key to the duality between varieties and covarieties and between equations and coequations developed in [83]. It is also crucial to strong

From this inverse we can construct a natural transformation $h : GL \to TG$, called its *mate* [69]. For our purposes, it is enough to say that h and G allow us to turn every L-algebra $\alpha : LA \to A$ into a T-coalgebra $h_A \circ G\alpha : GA \to GLA \to TGA$. We denote this operation $\widehat{G}(A, \alpha)$. In some sense, it is dual to the functor $\widehat{P}$ defined earlier. Given a T-coalgebra (X, γ), its *ultrafilter extension* is the T-coalgebra $\widehat{G}\widehat{P}(X, \gamma)$. A class C of T-coalgebras is *closed under ultrafilter extensions* if $(X, \gamma) \in \mathsf{C}$ implies $\widehat{G}\widehat{P}(X, \gamma) \in \mathsf{C}$. A class C of T-coalgebras *reflects ultrafilter extensions* if $\widehat{G}\widehat{P}(X, \gamma) \in \mathsf{C}$ implies $(X, \gamma) \in \mathsf{C}$.

▶ **Theorem 32** (Coalgebraic Goldblatt-Thomason Theorem [69]). *Let $T : \mathsf{Set} \to \mathsf{Set}$ preserve finite sets, and assume the existence of a natural inverse $\delta^{-1} : PT \to LP$ to the semantic transformation, then a class of T-coalgebras closed under ultrafilter extensions is definable by coequations-as-modal-formulas iff it is closed under homomorphic images, subcoalgebras, coproducts, and if it reflects ultrafilter extensions.*

7 Conclusion

In this review we have presented four types of syntaxes for "writing a coequation": *coequations-as-corelations*, which come with the special syntax of *coequations-as-equations* for functors of the type $\prod_{i \in I} A_i \times \mathsf{P}_i^{B_i}$ where each P_i is polynomial, and *coequations-as-predicates*, which come with the special syntax of *coalgebraic modal logic*. It is worth emphasising that the corelation and the predicate perspective are semantically equivalent and one can move from one to the other by taking an equalizer or a cokernel pair respectively. However, both the *syntax* and the *intuition* are different, and these aspects matter a great deal in practice.

A rule of thumb for which syntax to use in which situation might be the following. Thinking of the elements of a cofree coalgebra as generalized trees, if the aim is to specify a behaviour defined by a relationship between a tree and (some of) its children, then the corelation perspective is probably the most useful. This perspective was illustrated in §5, but can also be found in the beautiful work on stream differential equations of Hansen, Kupke and Rutten [46]. On the other hand, if the aim is to enforce or avoid a particular *behavioural pattern*, then the predicate perspective is probably the most useful.

Although much of our discussion focused on literature written decades ago, coequations continue to find new uses. It was recently observed that coequations appear in formal language theory as *varieties of languages* [13, 82], which play a dual role to monoid equations. A vastly wider perspective on this relationship was explored in subsequent work [81, 10]. For another example, a behavioural coequation appeared in a proof of the completeness of an axiomatisation of *guarded Kleene algebra with tests* (GKAT) [88], an algebraic framework for reasoning about simple imperative programs. There, the coequation is the set of behaviours specified by terms in the expression language of GKAT, much like the coequation in Example 10, and is used to present the covariety of automata that implement GKAT programs. This usage of coequations may also be possible in the context of an open problem posed by Milner [72], as a covariety implicitly appears in a recent partial solution [40].

As the examples above illustrate, the use value of coequations is emerging, slowly, from the literature. Given the scope of their applications, we hope that our synthesis of the literature will make coequations more accessible to the general computer science community.

completeness proofs in coalgebraic modal logic [25, 26].

References

1 Peter Aczel. *Non-Well-Founded Sets*. Number 14 in Lecture Notes. Center for the Study of Language and Information (CSLI), 1988.
2 Peter Aczel and Nax Mendler. A final coalgebra theorem. In *Category theory and computer science*, pages 357–365. Springer, 1989.
3 Jiří Adámek. Birkhoff's covariety theorem without limitations. *Commentationes Mathematicae Universitatis Carolinae*, 46(2):197–215, 2005.
4 Jiří Adámek. A logic of coequations. In *International Workshop on Computer Science Logic*, pages 70–86. Springer, 2005.
5 Jiří Adámek, Horst Herrlich, and George E Strecker. Abstract and concrete categories. The joy of cats, 2004.
6 Jiří Adámek and Hans-E Porst. From varieties of algebras to covarieties of coalgebras. *Electronic Notes in Theoretical Computer Science*, 44(1):27–46, 2001.
7 Jiří Adámek and Hans-E Porst. On varieties and covarieties in a category. *Mathematical Structures in Computer Science*, 13(2):201, 2003.
8 Jirí Adámek and Hans-E. Porst. On tree coalgebras and coalgebra presentations. *Theor. Comput. Sci.*, 311(1-3):257–283, 2004.
9 Jiří Adámek and Vera Trnková. *Automata and algebras in categories*, volume 37. Springer Science & Business Media, 1990.
10 Jiří Adámek, Stefan Milius, Robert S.R. Myers, and Henning Urbat. Generalized Eilenberg theorem: Varieties of languages in a category. *ACM Trans. Comput. Logic*, 20(1), 2018.
11 Steve Awodey and Jesse Hughes. The coalegebraic dual of Birkoff's variety theorem. Technical report, Carnegie Mellon University, 2000.
12 Adriana Balan, Alexander Kurz, and Jiří Velebil. Positive fragments of coalgebraic logics. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 51–65. Springer, 2013.
13 Adolfo Ballester-Bolinches, Enric Cosme-Llópez, and Jan J. M. M. Rutten. The dual equivalence of equations and coequations for automata. *Inf. Comput.*, 244:49–75, 2015.
14 Bernhard Banaschewski and Horst Herrlich. *Subcategories defined by implications*. McMaster Univ., 1975.
15 Hendrik Pieter Barendregt. *The lambda calculus - its syntax and semantics*, volume 103 of *Studies in logic and the foundations of mathematics*. North-Holland, 1985.
16 Michael Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114(2):299–315, 1993.
17 Garrett Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge*, 1935.
18 Patrick Blackburn, Johan FAK van Benthem, and Frank Wolter. *Handbook of modal logic*. Elsevier, 2006.
19 Janusz A. Brzozowski. Derivatives of regular expressions. *J. ACM*, 11(4):481–494, 1964.
20 Corina Cîrstea. A coequational approach to specifying behaviours. *Electronic Notes in Theoretical Computer Science*, 19:142–163, 1999.
21 Corina Cîrstea, Alexander Kurz, Dirk Pattinson, Lutz Schröder, and Yde Venema. Modal logics are coalgebraic. *The Computer Journal*, 54(1):31–41, 2011.
22 Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.
23 Ranald Clouston and Robert Goldblatt. Covarieties of coalgebras: comonads and coequations. In *International Colloquium on Theoretical Aspects of Computing*, pages 288–302. Springer, 2005.
24 John Horton Conway. *Regular algebra and finite machines*. Courier Corporation, 2012.
25 Fredrik Dahlqvist. *Completeness-via-canonicity for coalgebraic logics*. PhD thesis, Imperial College London, 2015 .
26 Fredrik Dahlqvist. Coalgebraic completeness-via-canonicity. In *International Workshop on Coalgebraic Methods in Computer Science*, pages 174–194. Springer, 2016.

27 Fredrik Dahlqvist and Alexander Kurz. The positivication of coalgebraic logics. In *7th Conference on Algebra and Coalgebra in Computer Science (CALCO)*, 2017.

28 Fredrik Dahlqvist and David Pym. Completeness via canonicity for distributive substructural logics: a coalgebraic perspective. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 119–135. Springer, 2015.

29 Robert Davis. Universal coalgebra and categories of transition systems. *Mathematical systems theory*, 4(1):91–95, 1970.

30 Robert Davis. Multivalued operations and universal coalgebra. *Proceedings of the American Mathematical Society*, 32(2):385–388, 1972.

31 Robert Davis. Quasi cotripleable categories,. *Proceedings of the American Mathematical Society*, 35:43–48, 1972.

32 Robert Davis. Combinatorial examples in universal coalgebra. *Proceedings of the American Mathematical Society*, 89(1):32–34, 1983.

33 Robert Davis. Combinatorial examples in universal coalgebra. iii. *Proceedings of the American Mathematical Society*, 92(3):332–334, 1984.

34 E Allen Emerson and Joseph Y Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of computer and system sciences*, 30(1):1–24, 1985.

35 W. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2013.

36 Nate Foster, Dexter Kozen, Matthew Milano, Alexandra Silva, and Laure Thompson. A coalgebraic decision procedure for NetKAT. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015*, pages 343–355. ACM, 2015.

37 Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *International Conference on Protocol Specification, Testing and Verification*, pages 3–18. Springer, 1995.

38 Robert Goldblatt. *Logics of time and computation*. Center for the Study of Language and Information, 1987.

39 Robert Goldblatt. A comonadic account of behavioural covarieties of coalgebras. *Mathematical Structures in Computer Science*, 15(2):243–269, 2005.

40 Clemens Grabmayer and Wan Fokkink. A complete proof system for 1-free regular expressions modulo bisimilarity. *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, July 2020.

41 H Peter Gumm. Elements of the general theory of coalgebras, 1999.

42 H Peter Gumm. Functors for coalgebras. *Algebra universalis*, 45(2):135–147, 2001.

43 H Peter Gumm and Tobias Schröder. Products of coalgebras. *Algebra Universalis*, 46(1):163–185, 2001.

44 Heinz-Peter Gumm. Equational and implicational classes of coalgebras. *Theoretical Computer Science*, 260(1-2):57–69, 2001.

45 Heinz-Peter Gumm and Tobias Schröder. Covarieties and complete covarieties. *Electronic Notes in Theoretical Computer Science*, 11:42–55, 1998.

46 Helle Hvid Hansen, Clemens Kupke, and Jan Rutten. Stream differential equations: specification formats and solution methods. *arXiv preprint arXiv:1609.08367*, 2016.

47 Ulrich Hensel and Horst Reichel. Defining equations in terminal coalgebras. In *Recent Trends in Data Type Specification*, pages 307–318. Springer, 1994.

48 Jesse Hughes. Modal operators for coequations. *Electronic Notes in Theoretical Computer Science*, 44(1):205–226, 2001.

49 Jesse Hughes. *A study of categories of algebras and coalgebras*. PhD thesis, Carnegie Mellon University, 2001.

50 Bart Jacobs. Mongruences and cofree coalgebras. In *International Conference on Algebraic Methodology and Software Technology*, pages 245–260. Springer, 1995.

51 Bart Jacobs. The temporal logic of coalgebras via Galois algebras. *Mathematical Structures in Computer Science*, 12(6):875–903, 2002.

52 Bart Jacobs. *Introduction to Coalgebra*, volume 59. Cambridge University Press, 2017.

53 Bart Jacobs and Ana Sokolova. Exemplaric expressivity of modal logics. *Journal of logic and computation*, 20(5):1041–1068, 2010.

54 Peter Jipsen. Concurrent Kleene algebra with tests. In Peter Höfner, Peter Jipsen, Wolfram Kahl, and Martin Eric Müller, editors, *Relational and Algebraic Methods in Computer Science - 14th International Conference, RAMiCS 2014. Proceedings*, volume 8428 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2014.

55 Christian Jäkel. A unified categorical approach to graphs, 2015. `arXiv:1507.06328`.

56 Tobias Kappé, Paul Brunet, Alexandra Silva, and Fabio Zanasi. Concurrent Kleene algebra: Free model and completeness. *CoRR*, abs/1710.02787, 2017.

57 Yasuo Kawahara and Masao Mori. A small final coalgebra theorem. *Theor. Comput. Sci.*, 233(1-2):129–145, 2000.

58 Dexter Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91)*, pages 214–225. IEEE Computer Society, 1991.

59 Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In *Computer Science Logic, 10th International Workshop, CSL '96, Annual Conference of the EACSL, 1996*, volume 1258 of *LNCS*, pages 244–259. Springer, 1996.

60 Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Algebraic semantics for coalgebraic logics. *Electronic Notes in Theoretical Computer Science*, 106:219–241, 2004.

61 Clemens Kupke, Alexander Kurz, and Dirk Pattinson. Ultrafilter extensions for coalgebras. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 263–277. Springer, 2005.

62 Clemens Kupke and Raul Andres Leal. Characterising behavioural equivalence: Three sides of one coin. In *Algebra and Coalgebra in Computer Science, Third International Conference, CALCO 2009. Proceedings*, volume 5728 of *LNCS*, pages 97–112. Springer, 2009.

63 Alexander Kurz. A co-variety-theorem for modal logic. *Advances in Modal Logic*, 2:367–380, 1998.

64 Alexander Kurz. Specifying coalgebras with modal logic. *Electronic Notes in Theoretical Computer Science*, 11:56–70, 1998.

65 Alexander Kurz. *Logics for coalgebras and applications to computer science*. PhD thesis, Ludwig-Maximilians-Universität München, 2000.

66 Alexander Kurz. Modal rules are co-implications. *Electronic Notes in Theoretical Computer Science*, 44(1):241–253, 2001.

67 Alexander Kurz. Specifying coalgebras with modal logic. *Theoretical Computer Science*, 260(1-2):119–138, 2001.

68 Alexander Kurz and Jirí Rosickỳ. Operations and equations for coalgebras. *Mathematical Structures in Computer Science*, 15(1):149, 2005.

69 Alexander Kurz and Jiří Rosickỳ. The Goldblatt-Thomason theorem for coalgebras. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 342–355. Springer, 2007.

70 Giulio Manzonetto and Antonino Salibra. From lambda-calculus to universal algebra and back. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008. Proceedings*, volume 5162 of *LNCS*, pages 479–490. Springer, 2008.

71 Michal Marvan. On covarieties of coalgebras. *Archivum Mathematicum*, 21(1):51–63, 1985.

72 Robin Milner. A complete inference system for a class of regular behaviours. *J. Comput. Syst. Sci.*, 28(3):439–466, 1984.

73 Lawrence S Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1-3):277–317, 1999.

74 Dirk Pattinson. Coalgebraic modal logic: Soundness, completeness and decidability of local consequence. *Theoretical Computer Science*, 309(1-3):177–193, 2003.

75 Boris I. Plotkin and Tanya Plotkin. Universal Algebra and Computer Science. In *Fundamentals of Computation Theory, 13th International Symposium, FCT 2001. Proceedings*, volume 2138 of *LNCS*, pages 35–44. Springer, 2001.

76 Horst Reichel. An approach to object semantics based on terminal co-algebras. *Mathematical Structures in Computer Science*, 5(2):129–152, 1995.

77 Grigore Roşu. A Birkhoff-like axiomatizability result for hidden algebra and coalgebra. *Electronic Notes in Theoretical Computer Science*, 11:176–193, 1998.

78 Grigore Roşu. Equational axiomatizability for coalgebra. *Theoretical Computer Science*, 260(1-2):229–247, 2001.

79 Jan JMM Rutten. Universal coalgebra: a theory of systems. Technical report, CWI, 1996.

80 Jan JMM Rutten. Universal coalgebra: a theory of systems. *Theoretical computer science*, 249(1):3–80, 2000.

81 Julian Salamanca. Unveiling Eilenberg-type correspondences: Birkhoff's theorem for (finite) algebras + duality. *CoRR*, abs/1702.02822, 2017.

82 Julian Salamanca, Adolfo Ballester-Bolinches, Marcello M. Bonsangue, Enric Cosme-Llópez, and Jan J. M. M. Rutten. Regular varieties of automata and coequations. In *Mathematics of Program Construction - 12th International Conference, MPC 2015*, volume 9129 of *LNCS*, pages 224–237. Springer, 2015.

83 Julian Salamanca, Marcello Bonsangue, and Jurriaan Rot. Duality of equations and coequations via contravariant adjunctions. In Ichiro Hasuo, editor, *Coalgebraic Methods in Computer Science*, pages 73–93, Cham, 2016. Springer International Publishing.

84 Antonino Salibra. On the algebraic models of lambda calculus. *Theor. Comput. Sci.*, 249(1):197–240, 2000.

85 Antonino Salibra and Robert Goldblatt. A finite equational axiomatization of the functional algebras for the lambda calculus. *Inf. Comput.*, 148(1):71–130, 1999.

86 Arto Salomaa. Two complete axiom systems for the algebra of regular events. *J. ACM*, 13(1):158–169, 1966.

87 Hanamantagouda P Sankappanavar and Stanley Burris. *A course in universal algebra*, volume 78. Citeseer, 1981.

88 Todd Schmid, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene algebra with tests: Coequations, coinduction, and completeness. *CoRR*, abs/2102.08286, 2021.

89 Daniel Schwencke. Coequational logic for finitary functors. *Electronic Notes in Theoretical Computer Science*, 203(5):243–262, 2008.

90 Daniel Schwencke. Coequational logic for accessible functors. *Information and Computation*, 208(12):1469–1489, 2010.

91 Steffen Smolka, Nate Foster, Justin Hsu, Tobias Kappé, Dexter Kozen, and Alexandra Silva. Guarded Kleene Algebra with Tests: Verification of Uninterpreted Programs in Nearly Linear Time. *CoRR*, abs/1907.05920, 2019.

92 Moshe Y Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 322–331. IEEE Computer Society, 1986.

93 Yde Venema. Algebras and coalgebras. In Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*, pages 331–426. North-Holland, 2007.

94 Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1992.

95 Uwe Wolter. On corelations, cokernels, and coequations. *Electronic Notes in Theoretical Computer Science*, 33:317–336, 2000.

96 James Worrell. Terminal sequences for accessible endofunctors. In Bart Jacobs and Jan J. M. M. Rutten, editors, *Coalgebraic Methods in Computer Science, CMCS 1999*, volume 19 of *Electronic Notes in Theoretical Computer Science*, pages 24–38. Elsevier, 1999.

Monads on Categories of Relational Structures

Chase Ford
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Stefan Milius
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Lutz Schröder
Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Abstract

We introduce a framework for universal algebra in categories of relational structures given by finitary relational signatures and finitary or infinitary Horn theories, with the arity λ of a Horn theory understood as a strict upper bound on the number of premisses in its axioms; key examples include partial orders ($\lambda = \omega$) or metric spaces ($\lambda = \omega_1$). We establish a bijective correspondence between λ-accessible enriched monads on the given category of relational structures and a notion of λ-ary *algebraic theories* (i.e. with operations of arity $< \lambda$), with the syntax of algebraic theories induced by the relational signature (e.g. inequations or equations-up-to-ϵ). We provide a generic sound and complete derivation system for such *relational algebraic theories*, thus in particular recovering (extensions of) recent systems of this type for monads on partial orders and metric spaces by instantiation. In particular, we present an ω_1-ary algebraic theory of metric completion. The theory-to-monad direction of our correspondence remains true for the case of κ-ary algebraic theories and κ-accessible monads for $\kappa < \lambda$, e.g. for finitary theories over metric spaces.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Logic and verification

Keywords and phrases monads, relational structures, Horn theories, relational logic

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.14

Related Version *Full Version*: https://arxiv.org/abs/2107.03880

Funding *Chase Ford*: Supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (393541319/GRK2475/1-2019).
Stefan Milius: Supported by the Deutsche Forschungsgemeinschaft (DFG) under project MI 717/7-1.
Lutz Schröder: Supported by Deutsche Forschungsgemeinschaft (DFG) under project SCHR 1118/15-1.

1 Introduction

Monads play an established role in the semantics of sequential and concurrent programming [21] – they encapsulate side-effects, such as statefulness, nontermination, nondeterminism, or probabilistic branching. The well-known correspondence between monads on the category of sets and algebraic theories [16] impacts accordingly on programming syntax, as witnessed, for example, in work on algebraic effects [24]: operations of the theory serve as syntax for computational effects such as non-deterministic or probabilistic choice. The comparative analysis of programs or systems beyond two-valued equivalence checking, e.g. under behavioural preorders, such as similarity, or behavioural distances, involves monads based on categories beyond sets, such as the categories **Pos** of partial orders or **Met** of (1-bounded) metric spaces. This has sparked recent interest in presentations of such monads using suitable variants of the notion of algebraic theory. While it is, in principle, possible to work with equational presentations that encapsulate the additional structure within the signature [14], it

seems at least equally natural to represent the additional structure (e.g. distance or ordering) within the judgements of the theory. Indeed, Mardare, Panangaden, and Plotkin replace equations with equations-up-to-ϵ in their *quantitative algebraic theories* [20], which present monads on **Met**, and in our own previous work on behavioural preorders [9] as well as in our own recent work with Adámek [5], we have used *in*equational theories to present monads on **Pos**.

In the present paper, we introduce a generalized approach to such notions of algebraic theory: We work in categories of finitary relational structures (more precisely, the objects are sets interpreting a given signature of finitary relation symbols), axiomatized by Horn theories whose axioms are implications with possibly infinite sets of antecedents. We say that such a theory is λ-ary for a regular cardinal λ if all its axioms have less than λ antecedents. For instance, **Pos** can be presented by a finitary (i.e. ω-ary) Horn theory over a binary relation $\leq$, and **Met** by an ω_1-ary Horn theory over binary relations $=_\epsilon$ "equality up to ϵ" indexed over rational numbers ϵ. We exploit that the models of a λ-ary Horn theory form a locally λ-presentable category $\mathscr{C}$ [4] to give a syntactic characterization of λ-accessible monads on $\mathscr{C}$ in terms of a notion of relational algebraic theory, in the sense that we prove a monad-theory correspondence. Following Kelly and Power [14], we use λ-presentable objects of $\mathscr{C}$ as arities and as contexts of axioms; however, as indicated above, we provide a syntax by expressing axioms using the relational signature instead of necessarily using only equality. We give a sound and complete deduction system for the arising *relational logic* (which generalizes standard equational logic), thus obtaining an explicit description of the monad generated by a relational algebraic theory in the indicated sense. One consequence of our main result is that quantitative algebraic theories [20] induce ω_1-accessible monads. More generally, presentations of ω_1-presentable monads in our formalism may involve operations with countable non-discrete arities: indeed, we present an ω_1-ary relational algebraic theory that defines the metric completion monad. We also take a glimpse at the more involved setting of κ-accessible monads on $\mathscr{C}$ where $\kappa < \lambda$ (e.g. finitary monads on **Met**). We give a partial characterization of κ-presentable objects in this setting, and show that while the monad-to-theory direction of our correspondence fails for $\kappa < \lambda$, the theory-to-monad direction does hold. This implies that some salient quantitative algebraic theories induce finitary monads; e.g. the theory of quantitative join-semilattices [20].

Related Work. We have already mentioned work by Kelly and Power on finitary monads [14] and by Mardare et al. on quantitative algebraic theories [20], as well as our own previous work [9] and our joint work with Adámek [5].

Power and Nishizawa [22] have extended the approach of Kelly and Power to deal with different enrichments of a category and the monads thereon, and obtain a correspondence between enriched Lawvere theories [25] and finitary enriched monads. More recently, Power and Garner [10] have provided a more thorough understanding of the equivalence between enriched finitary monads and enriched Lawvere theories as an instance of a free completion of an enriched category under a class of absolute colimits. Rosický [26] establishes a monad-theory correspondence for λ-accessible enriched monads and a notion of λ-ary enriched theory *á la* Linton [17], where arities of operations are given by pairs of objects. Like in the setting of Kelly and Power, relations (inequations, distances) are encoded in the arities. So the syntactic notion of theory is different from (and more abstract than) ours. Lucyshyn-Wright [18] establishes a rather general correspondence between monads and abstract theories in symmetric monoidal closed categories, parametric in a choice of arities, which covers several notions of theory and correspondences in the categorical literature under one roof.

Kurz and Velebil [15] characterize classical ordered varieties [6] (which are phrased in terms of inequalities) as precisely the exact categories in an enriched sense with a "suitable" generator. In recent subsequent work, Adámek at al. [2] establish a correspondence of these varieties with enriched monads on **Pos** that are strongly finitary [13], i.e. their underlying functor is a left Kan-extension of the embedding of finite discrete posets into **Pos**.

The main distinguishing feature of our work relative to the above is the explicit syntactic description of the monad obtained from a theory via a sound and complete derivation system. A related derivation system is the partial Horn logic of Palmgren and Vickers [23], which reasons about partial operations with unrestricted domains of definition. In contrast, we consider (varieties of) algebras with partial operations whose domain of definition is specified by objects in the base category (cf. Section 4); an essential ingredient in our monad-theory correspondence.

2 Preliminaries

We review the basic theory of locally presentable categories (see [4] for more detail) and of monads. We assume a modest familiarity with the elementary concepts of category theory [3] and with ordinal and cardinal numbers [11]. We write $\mathsf{card}\, X$ for the cardinality of a set X and, where λ is a cardinal, we write $X' \subseteq_\lambda X$ to indicate that $X' \subseteq X$ and $\mathsf{card}\, X' < \lambda$.

Locally Presentable Categories. Fix a regular cardinal λ (i.e. an infinite cardinal which is not cofinal to any smaller cardinal). A poset $(I, \leq)$ is λ-*directed* if each subset $I_0 \subseteq_\lambda I$ has an upper bound: there exists $u \in I$ such that $i \leq u$ for all $i \in I_0$. A λ-*directed diagram* is a functor whose domain is a λ-directed poset (viewed as a category); colimits of such diagrams are also called λ-*directed*. An object X in a category $\mathscr{C}$ is λ-*presentable* if the covariant hom-functor $\mathscr{C}(X, -)$ preserves λ-directed colimits. That is, X is λ-presentable if for each λ-directed colimit $(D_i \xrightarrow{c_i} C)_{i\in I}$ in $\mathscr{C}$, every morphism $m\colon X \to C$ factors through one of the c_i essentially uniquely: there exists $i \in I$ and $g\colon X \to D_i$ such that $m = c_i \cdot g$, and for all $g'\colon X \to D_i$ such that $m = c_i \cdot g'$, there exists $j \geq i$ such that $D(i \to j) \cdot g = D(i \to j) \cdot g'$.

▶ **Definition 2.1.** A category $\mathscr{C}$ is *locally λ-presentable* if it is cocomplete, its full subcategory $\mathsf{Pres}_\lambda(\mathscr{C})$ given by the λ-presentable objects of $\mathscr{C}$ is essentially small, and every $C \in \mathscr{C}$ is a λ-directed colimit of objects in $\mathsf{Pres}_\lambda(\mathscr{C})$. When $\lambda = \omega$ (resp. ω_1), we speak of *locally finitely (resp. countably) presentable categories*. We call $\mathscr{C}$ *locally presentable* if it is locally λ-presentable for some cardinal λ. A functor F on a locally presentable category is λ-*accessible* if it preserves λ-directed colimits. When $\lambda = \omega$ or ω_1, we speak of *finitary* and *countably accessible functors*, respectively.

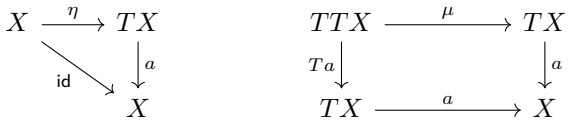
Reflective subcategories. A full subcategory $\mathscr{C}'$ of a category $\mathscr{C}$ is *reflective* if the embedding $\iota\colon \mathscr{C}' \hookrightarrow \mathscr{C}$ is a right adjoint. In this case, we write $r_X\colon X \to RX$ (or just r if X is clear from the context) for the universal arrows; we call RX *the reflection of* $X \in \mathscr{C}$, r_X the *reflective arrow*, and the left adjoint R the *reflector*. The universal property of $r_X\colon X \to RX$ is as follows: For each morphism $f\colon X \to Y$ in $\mathscr{C}$, where Y lies in $\mathscr{C}'$, there exists a unique morphism $f^\sharp\colon RX \to Y$ such that $f = f^\sharp \cdot r_X$. We call $\mathscr{C}'$ *epi-reflective* if r_X is epi for all $X \in \mathscr{C}$. We will employ the following reflection theorem:

▶ **Theorem 2.2** [4, Cor. 2.48]**.** *If $\mathscr{C}'$ is a full subcategory of a locally λ-presentable category $\mathscr{C}$ and $\mathscr{C}'$ is closed under limits and λ-directed colimits in $\mathscr{C}$, then $\mathscr{C}'$ is reflective and locally λ-presentable.*

Monads. A *monad* on a category $\mathscr{C}$ is a functor $T\colon \mathscr{C} \to \mathscr{C}$ equipped with natural transformations $\eta\colon \mathsf{Id} \to T$ (the *unit*) and $\mu\colon TT \to T$ (the *multiplication*) such that the diagrams below commute.

We call the monad (T, η, μ) λ-*accessible* if its underlying functor is λ-accessible.

▶ **Definition 2.3.** An *Eilenberg-Moore algebra* for the monad (T, η, μ) is a $\mathscr{C}$-morphism of the shape $a\colon TX \to X$ satisfying the following coherence laws:



A *homomorphism* from $a\colon TX \to X$ to an Eilenberg-Moore algebra $b\colon TY \to Y$ is a morphism $h\colon X \to Y$ in $\mathscr{C}$ such that $h \cdot a = Th \cdot b$.

▶ **Notation 2.4.** For a functor $F\colon \mathscr{C} \to \mathscr{C}$, we write $\mathsf{Alg}\, F$ for the category of F-algebras and homomorphisms, i.e. $\mathsf{Alg}\, F$ has $\mathscr{C}$-morphisms of the shape $a\colon FA \to A$ as objects, and a homomorphism $(A, a) \to (B, b)$ is a $\mathscr{C}$-morphism $h\colon A \to B$ such that $h \cdot a = b \cdot Fh$.

3 Categories of Relational Structures

As indicated previously, we will study monads over base categories consisting of (single-sorted) relational structures. Specifically, we will restrict the relational signature to be finitary but allow infinitary Horn axioms. We proceed to recall basic definitions, examples, and results, in particular on closed structure and (local) presentability. In Section 3.1, we present new results on the partial characterization of (internally) λ-presentable objects in cases where the overall local presentability index of the category is greater than λ.

▶ **Definition 3.1.**

1. A *relational signature* is a set Π of *relation symbols* $\alpha, \beta, \ldots$ together with a finite *arity* $0 < \mathsf{ar}(\alpha) \in \omega$ for all $\alpha \in \Pi$. A Π-*edge* in a set S is a pair $e = \alpha(f)$ where $\alpha \in \Pi$ and $f\colon \mathsf{ar}(\alpha) \to S$ is a function. For a map $g\colon S \to Y$, we write $g \cdot e = \alpha(g \cdot f)$. We extend this notation pointwise to sets E of edges: $g \cdot E = \{g \cdot e \mid e \in E\}$.
2. A Π-*structure* X consists of an underlying set $|X|$ (or just X when no confusion is likely) and a set $\mathsf{E}(X)$ of Π-edges in $|X|$. If $\alpha(f) \in \mathsf{E}(X)$, we write $\alpha_X(f)$ or even $X \models \alpha(f)$.
3. A *relation-preserving map* (or briefly a *morphism*) from X to a Π-structure Y is a function $g\colon |X| \to |Y|$ such that $g \cdot \mathsf{E}(X) \subseteq \mathsf{E}(Y)$. We call g an *embedding* if g is injective and *relation-reflecting*, i.e. if $Y \models g \cdot e$ for an edge $e = (\alpha, f\colon \mathsf{ar}(\alpha) \to X)$, then $X \models e$. We denote by $\mathbf{Str}(\Pi)$ the category of Π-structures and relation-preserving maps.

▶ **Notation 3.2.** Given an edge $\alpha(f)$ such that $f(i) := x_i$ for all $i \in \mathsf{ar}(\alpha)$, we sometimes write $\alpha(x_1, \ldots, x_{\mathsf{ar}(\alpha)})$ or even $\alpha(x_i)$ in lieu of $\alpha(f)$. We will pass between these presentations without further mention.

We are going to carve out full subcategories of $\mathbf{Str}(\Pi)$ by means of infinitary Horn axioms, whose **syntax** we recall next.

▶ **Definition 3.3.** Let Π be a relational signature, and λ a regular cardinal. We fix a set Var of variables such that $\mathsf{card}(\mathsf{Var}) = \lambda$. A *$\lambda$-ary Horn formula* over Π has the form

$$\Phi \implies \psi$$

where Φ is a set of Π-edges in Var such that $\mathsf{card}\,\Phi < \lambda$ and ψ is a $\Pi \cup \{=\}$-edge in Var, for a fresh binary relation symbol $=$. In case $\Phi = \{\varphi_1, \dots, \varphi_n\}$ is finite, we write $\varphi_1, \dots, \varphi_n \implies \psi$, and if $\Phi = \emptyset$, then we just write $\implies \psi$. A *λ-ary Horn theory* $\mathscr{H} = (\Pi, \mathcal{A})$ consists of a relational signature Π and a set $\mathcal{A}$ of λ-ary Horn formulae over Π, the *axioms* of $\mathscr{H}$.

We fix a λ-ary Horn theory $\mathscr{H} = (\Pi, \mathcal{A})$ for the rest of the paper. We define the **semantics** of Horn formulae in a Π-structure X as follows. We denote by $\overline{X}$ the $\Pi \sqcup \{=\}$-structure obtained from X by putting $=_X := \{(x, x) \mid x \in X\}$. A *valuation* is a map $\kappa\colon \mathsf{Var} \to |X|$. We say that X *satisfies* a Horn formula $\Phi \implies \psi$ if whenever κ is a valuation such that $X \models \kappa \cdot \phi$ for all $\phi \in \Phi$, then $\overline{X} \models \kappa \cdot \psi$. Finally, X is a *model* of $\mathscr{H}$, or of $\mathcal{A}$, if X satisfies all axioms of $\mathscr{H}$. The full subcategory of $\mathbf{Str}(\Pi)$ spanned by the models of $\mathcal{A}$ is $\mathbf{Str}(\Pi, \mathcal{A})$ (or $\mathbf{Str}\,\mathscr{H}$).

We have an obvious notion of *derivation* under $\mathscr{H}$ *over* a given set Z (e.g. of variables or points in a structure): We extend $\mathscr{H}$ to $(\Pi \cup \{=\}, \bar{\mathcal{A}})$ where $\bar{\mathcal{A}}$ consists of the axioms in $\mathcal{A}$ and additional axioms stating that $=$ is an equivalence and that all relations in Π are closed under $=$ in the obvious sense. Then we have a single (λ-ary) derivation rule for application of Horn axioms $(\Phi \implies \psi) \in \bar{\mathcal{A}}$ over Z:

$$\frac{\kappa \cdot \Phi}{\kappa \cdot \psi}\ (\kappa\colon \mathsf{Var} \to Z).$$

We say that a set E of edges over Z *entails* an edge e *over* Z (*under* $\mathscr{H}$) if e is derivable from edges in E in this system. In case $Z = \mathsf{Var}$ and $\mathsf{card}\,E < \lambda$, the expression $E \implies e$ is in fact a Horn formula, and we then also say that $\mathscr{H}$ *entails* $E \implies e$ if E entails e.

▶ **Assumption 3.4.** For technical convenience, we assume that the fixed Horn theory $\mathscr{H} = (\Pi, \mathcal{A})$ *expresses equality*. That is, there exists a set $Eq(x, y)$ of Π-edges in variables x, y such that $\mathscr{H}$ entails $Eq(x, y) \implies x = y$ as well as $\implies \psi$ for all edges $\psi \in Eq(x, x)$ (where we use obvious notation for substitution; formally, $Eq(x, x) = g \cdot Eq(x, y)$ where $g(x) = g(y) = x$). Moreover, we assume that $\mathcal{A}$ explicitly includes the (derivable) formulae $Eq(x_1, y_1) \cup \dots \cup Eq(x_{\mathsf{ar}(\alpha)}, y_{\mathsf{ar}(\alpha)}) \cup \{\alpha(x_1, \dots, x_{\mathsf{ar}(\alpha)})\} \implies \alpha(y_1, \dots, y_{\mathsf{ar}(\alpha)})$ saying that all relations $\alpha \in \Pi$ are closed under Eq (implying also that Eq is symmetric and transitive). This is without loss of generality as we can always extend a given Horn theory with an equality predicate axiomatized by the above conditions without changing its category of models; indeed we leave this predicate implicit in examples whose natural presentation does not include it.

▶ **Example 3.5.** We mention some key examples of Horn theories:

1. The category **Set** of sets and functions is specified by the trivial Horn theory $(\emptyset, \emptyset)$.
2. The category **Pos** of partially ordered sets (posets) and monotone maps is specified by the ω-ary Horn theory consisting of a single binary relation symbol $\leq$ and the axioms

$$x \leq x; \qquad x \leq y,\ y \leq z \implies x \leq z; \qquad \text{and} \qquad x \leq y,\ y \leq x \implies x = y.$$

This theory expresses equality (Assumption 3.4) via $Eq(x, y) = \{x \leq y, y \leq x\}$.

3. The theory $\mathscr{H}_{\mathbf{Met}}$ of *metric spaces* is the ω_1-ary theory consisting of binary relation symbols $=_\epsilon$ for all $\epsilon \in \mathbb{Q} \cap [0,1]$, and the axioms

$$\implies x =_0 x \qquad (\textbf{Refl})$$
$$x =_0 y \implies x = y \qquad (\textbf{Equal})$$
$$x =_\epsilon y \implies y =_\epsilon x \qquad (\textbf{Sym})$$
$$x =_\epsilon y, y =_{\epsilon'} z \implies x =_{\epsilon+\epsilon'} z \qquad (\textbf{Triang})$$
$$x =_\epsilon y \implies x =_{\epsilon+\epsilon'} y \qquad (\textbf{Up})$$
$$\{x =_{\epsilon'} y \mid \mathbb{Q}_{\geq 0} \ni \epsilon' > \epsilon\} \implies x =_\epsilon y \qquad (\textbf{Arch})$$

   where ϵ, ϵ' range over $\mathbb{Q} \cap [0,1]$ (that is, the axioms mentioning ϵ, ϵ' are in fact axiom schemes representing one axiom for each ϵ, ϵ'). This theory expresses equality via $Eq(x,y) = \{x =_0 y\}$; in fact, even if we remove $=_0$, the remaining theory still expresses equality via $Eq(x,y) = \{x =_{1/n} y \mid n > 0\}$. The theory $\mathscr{H}_{\mathbf{Met}}$ specifies the category **Met** of 1-bounded metric spaces and non-expansive maps, in the sense that $\mathbf{Str}(\mathscr{H}_{\mathbf{Met}})$ and **Met** are concretely isomorphic: $X \in \mathbf{Str}(\mathscr{H}_{\mathbf{Met}})$ induces the 1-bounded metric space (X,d) given by $d(x,y) = \bigwedge\{\epsilon \mid x =_\epsilon y \in \mathsf{E}(X)\}$, and conversely a metric space (X,d) induces the $\mathscr{H}_{\mathbf{Met}}$-model on X with edges $\{x =_\epsilon y \mid x,y \in X, d(x,y) \leq \epsilon\}$.
4. Consider the theory obtained by taking two "copies" of the theory $\mathscr{H}_{\mathbf{Met}}$: its signature consists of binary relation symbols $=^0_\epsilon, =^1_\epsilon$ for all $\epsilon \in \mathbb{Q} \cap [0,1]$, each subject to (indexed variants of) the axiom schema above. This yields an ω_1-ary theory of *bi-metric spaces*: sets equipped with a pair of metrics. Morphisms are maps which are non-expansive with respect to both metrics. Further imposing axioms of the shape

$$x =^0_\epsilon y \implies x =^1_\epsilon y \qquad (\epsilon \in \mathbb{Q} \cap [0,1])$$

   specifies bi-metric spaces in which one metric is always finer than the other. We aim to approach the problem of digital fingerprinting in future work on graded monads in precisely this setting.
5. Let L be a complete lattice (for simplicity), and let $L_0 \subseteq L$ be meet-dense in L in the sense that $l = \bigwedge\{p \in L_0 \mid p \geq l\}$ for each $l \in L$; whenever $q \geq \bigwedge P$ for $q \in L_0$ and $P \subseteq L_0$ such that $\bigwedge P \notin L_0$, then $q \geq p$ for some $p \in P$ (e.g. these conditions hold trivially for $L_0 = L$). Further, fix λ such that $|L_0| < \lambda$. Let $\mathscr{H}_L$ be the λ-ary Horn theory with binary relation symbols α_p for all $p \in L_0$ and axioms

$$\{\alpha_p(x,y) \mid p \in P\} \implies \alpha_q(x,y) \qquad (P \subseteq L_0, q = \textstyle\bigwedge P \in L_0) \qquad (\textbf{Arch})$$
$$\alpha_p(x,y) \implies \alpha_q(x,y) \qquad (p,q \in L_0, p \leq q) \qquad (\textbf{Up})$$

   where p,q range over L_0. Then $\mathbf{Str}(\mathscr{H}_L)$ is concretely isomorphic to the category of *L-valued relations*, whose objects X are sets X equipped with map $P\colon X \times X \to L$, and whose morphisms $(X,P) \to (Y,Q)$ are maps $X \to Y$ such that $Q(f(x),f(y)) \leq P(x,y)$. (Of course, **Met** is essentially the special case $L = [0,1]$, $L_0 = \mathbb{Q} \cap [0,1]$ with some additional axioms.)
6. A signature of partial operations is a set P of operation symbols f with assigned finite arities $\mathsf{ar}(f)$. A (partial) P-algebra is then a set A and, for each $f \in P$, a partial function $f_A\colon A^{\mathsf{ar}(f)} \to A$. A homomorphism of partial algebras is a map $h\colon A \to B$ such that whenever $f_A(x_1,\dots,x_{\mathsf{ar}(f)})$ is defined, then $f_B(h(x_1),\dots,h(x_{\mathsf{ar}(f)}))$ is defined and equals $h(f_A(x_1,\dots,x_{\mathsf{ar}(f)}))$. The category of partial P-algebras and their homomorphims is concretely isomorphic to the category of models of the ω-ary Horn theory consisting of

relational symbols α_f of arity $\mathsf{ar}(f)+1$ for all $f \in P$ (with $\alpha_f(x_1,\dots,x_{\mathsf{ar}(f)},y)$ being understood as $f(x_1,\dots,x_{\mathsf{ar}(f)}) = y$), and axioms

$$\alpha_f(x_1,\dots,x_{\mathsf{ar}(f)},y), \alpha_f(x_1,\dots,x_{\mathsf{ar}(f)},z) \implies y = z.$$

We proceed to discuss some key aspects of the categorical structure of $\mathbf{Str}(\mathscr{H})$.

Reflection. We first note

▶ **Proposition 3.6.** $\mathbf{Str}(\Pi,\mathcal{A})$ *is a (full) epi-reflective subcategory of* $\mathbf{Str}(\Pi)$.

Since $\mathbf{Str}(\Pi)$ is easily seen to be complete and cocomplete, it follows that $\mathbf{Str}(\Pi,\mathcal{A})$ is cocomplete and moreover closed under limits in $\mathbf{Str}(\Pi)$, and hence complete. We write

$$R\colon \mathbf{Str}(\Pi) \to \mathbf{Str}(\Pi,\mathcal{A}) \quad \text{and} \quad r_X\colon X \to RX$$

for the left adjoint of the inclusion $\mathbf{Str}(\Pi,\mathcal{A}) \hookrightarrow \mathbf{Str}(\Pi)$ (the *reflector*) and the corresponding (surjective) reflection maps, respectively. Explicitly, RX is constructed as follows. We define an equivalence $\sim$ on X by $x \sim y$ if $\mathsf{E}(X)$ entails $x = y$ under $\mathscr{H}$ (in the sense defined above), and let $q\colon X \to X/{\sim}$ denote the quotient map; then RX has underlying set $X/{\sim}$, and contains precisely the edges $q \cdot e$ such that $\mathsf{E}(X)$ entails e; moreover, $r_X = q$ as a map.

Local presentability. One easily checks

▶ **Lemma 3.7.** *An object* $(X,E) \in \mathbf{Str}(\Pi)$ *is* λ*-presentable iff* $\mathsf{card}\, X < \lambda$ *and* $\mathsf{card}\, E < \lambda$*; the category* $\mathbf{Str}(\Pi)$ *is locally finitely presentable.*

By Proposition 3.6 and since $\mathbf{Str}(\Pi,\mathcal{A})$ is easily seen to be closed under λ-directed colimits in $\mathbf{Str}(\Pi)$, we thus have

▶ **Proposition 3.8** [4, Example 5.27(3)]**.** $\mathbf{Str}(\mathscr{H})$ *is locally* λ*-presentable.*

The forgetful functor $\mathbf{Str}(\mathscr{H}) \to \mathbf{Set}$ preserves λ-directed colimits. Moreover, we have an easy characterization of λ-presentable objects:

▶ **Proposition 3.9.** *For an* $\mathscr{H}$*-model* X*, the following are equivalent.*

1. X *is* λ*-presentable in* $\mathbf{Str}(\Pi,\mathcal{A})$*;*
2. $X \cong R(Y,E)$ *for some* λ*-presentable* $(Y,E) \in \mathbf{Str}(\Pi)$*;*
3. $\mathsf{card}\,|X| < \lambda$*, and* X *is* λ-generated*, i.e. there exists* $E \subseteq \mathsf{E}(X)$ *such that* $\mathsf{card}\, E < \lambda$ *and* E *entails every edge in* $\mathsf{E}(X)$ *under* $\mathscr{H}$ *(equivalently,* Ri *is an isomorphism where* $i\colon (|X|,E) \to X$ *is the* $\mathbf{Str}(\Pi)$*-morphism carried by* id_X*).*

▶ **Remark 3.10.** For instance, every finite partial order is ω-presentable, and every countable metric space is ω_1-presentable. We emphasize that the situation is more complicated for κ-presentable objects where $\kappa < \lambda$; we treat this case in more detail in Section 3.1. For instance, every finite metric space with rational distances (cf. Example 3.5) is finitely generated in the sense of Proposition 3.9 but not finitely presentable.

Closed monoidal structure. The point-wise structure defines an *internal hom* functor:

▶ **Definition 3.11.** The *internal hom of* $X, Y \in \mathbf{Str}(\Pi)$ is the Π-structure $[X,Y]$ carried by $\mathbf{Str}(\Pi)(X,Y)$ with the edge set defined by

$$\mathsf{E}([X,Y]) := \{e \mid \forall x \in X.\, \pi_x \cdot e \in \mathsf{E}(Y)\},$$

where $\pi_x \colon \mathbf{Str}(\Pi)(X,Y) \to Y$ is defined by $\pi_x(g) = g(x)$. For each $X \in \mathbf{Str}(\Pi)$, the assignment $Y \mapsto [X,Y]$ defines a (covariant) *internal hom functor*

$$[X,-] \colon \mathbf{Str}(\Pi) \to \mathbf{Str}(\Pi)$$

with the action on a morphism $m \colon Y \to Z$ given by post-composition: $[X,m](g) := m \cdot g$.

One can show that $[-,-]$ endows $\mathbf{Str}(\Pi)$ with a symmetric closed structure. Using results of Day and LaPlaza [8] it follows that $\mathbf{Str}(\Pi)$ is a symmetric monoidal closed category. The ensuing monoidal product is given by the structure $X \otimes Y$ with underlying set $X \times Y$ and edges

$$\{e \mid (\pi_1 \cdot e \text{ constant } \wedge \pi_2 \cdot e \in \mathsf{E}(Y)) \vee (\pi_2 \cdot e \text{ constant } \wedge \pi_1 \cdot e \in \mathsf{E}(X))\}.$$

where an edge (α, f) is *constant* if f is a constant map, and $\pi_1 \colon X \times Y \to X$ and $\pi_2 \colon X \times Y \to Y$ are the projection maps. The tensor unit is the trivial one point structure I_0 with no edges. To verify that this is the monoidal product arising from the symmetric closed structure $[-,-]$ it suffices to show that $(-) \otimes X$ is left adjoint to $[X,-]$. Indeed, we have $\mathbf{Str}(\Pi)$-morphisms $u_Y \colon Y \to [X, Y \otimes X]$, $u_Y(y) = \lambda x.(y,x)$. It is straightforward to check that u_Y is a universal arrow. That is:

▶ **Proposition 3.12.** *For every $X \in \mathbf{Str}(\Pi)$, $(-) \otimes X$ is a left adjoint of $[X,-]$.*

Moreover, one easily checks that $[X,-]$ restricts to a functor

$$[X,-] \colon \mathbf{Str}(\mathscr{H}) \to \mathbf{Str}(\mathscr{H}).$$

Hence, we can apply Day's reflection theorem [7, §1] to the reflector $R \colon \mathbf{Str}(\Pi) \to \mathbf{Str}(\Pi, \mathcal{A})$ (see the discussion immediately below Proposition 3.6) to obtain

▶ **Corollary 3.13.** *The category $\mathbf{Str}(\mathscr{H})$ is a closed symmetric monoidal category, with monoidal structure*

$$X \otimes_{\mathscr{H}} Y = R(X \otimes Y), \qquad I = RI_0$$

and internal hom given by $[X,-] \colon \mathbf{Str}(\mathscr{H}) \to \mathbf{Str}(\mathscr{H})$.

We briefly refer to $\otimes_{\mathscr{H}}$ as the *Manhattan product*.

▶ **Example 3.14.** 1. In **Pos** (Example 3.5.2), the Manhattan product coincides with binary Cartesian product (so **Pos** is Cartesian closed).
2. In **Met** (Example 3.5.3), the Manhattan product $(X, d_X) \otimes_{\mathscr{H}_{\mathbf{Met}}} (Y, d_Y)$ is $X \times Y$ equipped with the well-known Manhattan metric d given by $d((x_1,y_1),(x_2,y_2)) = \min(d_X(x_1,x_2) + d_Y(y_1,y_2), 1)$ (while Cartesian products carry the supremum metric).

▶ **Definition 3.15.** A functor $F \colon \mathbf{Str}(\mathscr{H}) \to \mathbf{Str}(\mathscr{H})$ that preserves the pointwise structure on morphisms is called *enriched*. That is, we call F enriched if for all $X, Y \in \mathbf{Str}(\mathscr{H})$ and all edges $f \colon \mathsf{ar}(\alpha) \to \mathbf{Str}(\mathscr{H})(X,Y)$ $(\alpha \in \Pi)$, if $[X,Y] \models \alpha(f_i)$, then $[FX,FY] \models \alpha(F(f_i))$.

Internal local presentability. For use of objects X as arities of operations, we will in fact need that the *internal* hom $[X,-]$ is λ-accessible. Using Kelly's results [12, (5.2) and (5.3)] this holds precisely for the λ-presentable objects since we have

▶ **Proposition 3.16.** *The λ-presentable objects of $\mathbf{Str}(\mathscr{H})$ are closed under the monoidal structure. That is, $I = RI_0$ is λ-presentable and $X \otimes_{\mathscr{H}} Y$ is λ-presentable whenever X and Y are so.*

In fact, this implies that $\mathbf{Str}(\mathscr{H})$ is *locally λ-presentable as a (symmetric monoidal) closed category* [12, (5.5)].

3.1 Compact Horn Models

We have seen above that the category $\mathbf{Str}(\mathscr{H})$ (where $\mathscr{H}$ is a λ-ary Horn theory) is (internally) locally λ-presentable, with a straightforward characterization of the (internally) λ-presentable objects (Propositions 3.9 and 3.16). We proceed to look at the rather less straightforward notion of internally κ-presentable objects in $\mathbf{Str}(\Pi, \mathcal{A})$ for $\kappa < \lambda$. The main scenario that motivates our interest in this case is that of finitary monads on categories that are internally locally λ-presentable only for some $\lambda > \omega$, such as metric spaces.

Further unfolding definitions, we have that an object X is internally κ-presentable if for every κ-directed colimit $(D_i \xrightarrow{c_i} C)_{i\in I}$, the canonical morphism

$$\mathsf{colim}[X, D_i] \to [X, \mathsf{colim}\, D_i]$$

is an isomorphism. We split this property into two parts: We say that X is *weakly κ-presentable* if the canonical morphism is always surjective, and *co-weakly κ-presentable* if the canonical morphism is always an embedding. Below, we give necessary and sufficient conditions for weak κ-presentability. Co-weak κ-presentability is a more elusive property; more concretely, it means roughly that X-indexed tuples of derivations in the given Horn theory can be synchronized into single derivations over X-indexed tuples of points. We give some examples below (Example 3.17).

▶ **Example 3.17.** We give some examples and non-examples of internally finitely presentable objects in locally ω_1-presentable categories $\mathbf{Str}(\Pi, \mathcal{A})$.

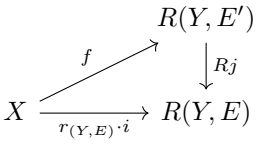
1. A metric space is internally finitely presentable iff it is finite and discrete. The "if" direction has surprisingly complicated reasons: It holds only because over the reals, finite joins distribute over directed infima. On the other hand, no non-empty metric space is *externally* finitely presentable, as its hom-functor will fail to preserve the colimit of the directed chain $(D_i)_{i<\omega}$ of spaces D_i with underlying set $\{0, 1\}$ and metric $d(0, 1) = 1/(i + 1)$.
2. In the category of L-valued relations for a complete lattice L in which binary joins fail to distribute over directed infima (such lattices exist), the two-element discrete space fails to be internally finitely presentable.
3. Let L be as in the previous item, and assume additionally that there is $l \in L$ such that in the downset of l, finite joins do distribute over directed infima (again, such L exist). Take the Horn theory of L-valued relations, extended with an additional (two-valued) relation α and axioms

$$\alpha(x, y) \wedge \alpha(x', y') \implies x =_l x' \qquad \alpha(x, y) \wedge \alpha(x', y') \implies y =_l y'.$$

   Then the set $\{0, 1\}$ equipped with the discrete L-valued relation and $\alpha(0, 1)$ is internally finitely presentable.

We proceed to give the announced characterization of weakly finitely presentable objects.

▶ **Definition 3.18.** A *cover* (Y, E), or just E, of $X \in \mathbf{Str}(\Pi, \mathcal{A})$ is a set E of edges in some set $Y \supseteq |X|$ such that all edges of X are implied by those in E under the Horn theory $\mathcal{A}$. That is, the underlying map $r_{(Y,E)} \colon Y \to |R(Y, E)|$ of the reflection composes with the inclusion $i \colon |X| \hookrightarrow Y$ to yield a morphism $r_{(Y,E)} \cdot i \colon X \to R(Y, E)$ (in $\mathbf{Str}(\Pi, \mathcal{A})$). Then X is *κ-compact* if for each cover (Y, E) of X there exist $E' \subseteq_\kappa E$ and a morphism $f \colon X \to R(Y, E')$ such that $r_{(Y,E)} \cdot i = Rj \cdot f$ where $j \colon (Y, E') \to (Y, E)$ is the $\mathbf{Str}(\Pi)$-morphism carried by id_Y:



(3.1)

▶ **Lemma 3.19.** *Every κ-compact object is κ-generated.*

▶ **Remark 3.20.** We will show that the weakly finitely presentable objects in $\mathbf{Str}(\Pi, \mathcal{A})$ are precisely the κ-compact objects with less than κ elements (Proposition 3.21). This characterization breaks under seemingly innocuous variations of the definition of κ-compactness:

1. It is essential that the edges of a cover live over a superset Y of X. If we were to restrict covers to consist of edges over X (call such a cover an *X-cover*), then finite ω-compact objects in the arising relaxed sense would in general fail to be finitely presentable. E.g. take $(\Pi, \mathcal{A})$ to be the theory of metric spaces additionally equipped with a transitive relation α. Then the set $X = \{0, 2\}$ equipped with the discrete metric and the edge $\alpha(0, 2)$ satisfies the relaxed definition of compactness (every X-cover must contain the edge $\alpha(0, 2)$) but fails to be weakly finitely presentable: The colimit of the ω-chain of objects D_i with underlying set $\{0, 1, 1', 2\}$, distances $d(0, 1) = d(1', 2) = 1$, $d(1, 1') = 1/i$, and edges $\alpha(0, 1)$ and $\alpha(1', 2)$ is not weakly preserved by the hom-functor $\mathbf{Str}(\Pi, \mathcal{A})(X, -)$ (the obvious inclusion of X into the colimit fails to factorize through any of the D_i).
2. Note that we do not require that the factorization f of $r_{(Y,E)} \cdot i$ in (3.1) equals $r_{(Y,E')} \cdot i$; i.e. f may rename elements of X into elements of Y that lie outside X. Let us refer to the natural-sounding strengthening of κ-compactness where we do require $f = r_{(Y,E')} \cdot i$ as *strong* κ-compactness; e.g. X is strongly ω-compact if every cover of X has a finite subcover. However, this notion is too strong, i.e. not every (weakly) finitely presentable object in $\mathbf{Str}(\Pi, \mathcal{A})$ is strongly ω-compact. As a counterexample, consider the same Horn theory as in the previous item but without the transitivity axiom for α. Then the same object X as in the previous item is weakly finitely presentable (even internally finitely presentable) but not strongly ω-compact, as witnessed by the cover $E = \{\alpha(0', 2)\} \cup \{0 =_{1/n} 0' \mid n > 0\}$.

▶ **Proposition 3.21.** *The following are equivalent for $X \in \mathbf{Str}(\Pi, \mathcal{A})$:*

1. *X is weakly κ-presentable;*
2. *X is κ-compact, and* $\mathsf{card}\,|X| < \kappa$.

4 Relational Algebraic Theories

We next describe a framework of universal algebra for enriched κ-accessible monads on the internally locally λ-presentable category $\mathbf{C} = \mathbf{Str}(\mathscr{H})$ of $\mathscr{H}$-models, for $\kappa \leq \lambda$. We etablish one direction of our theory-monad correspondence: We show that every theory in our framework induces a κ-accessible monad (Remark 4.12) whose algebras are precisely the models of the theory (Theorem 4.13). We address the converse direction in Section 5. We write $\mathbf{C}_0$ for the ordinary category underlying the closed monoidal category $\mathbf{C}$.

Following Kelly and Power [14], we use the internally λ-presentable objects in $\mathbf{C}$ as the arities of operation symbols. The full subcategory $\mathsf{Pres}_\lambda(\mathbf{C})$ of internally λ-presentable objects is essentially small (Proposition 3.16); *we fix a small subcategory $\mathscr{P}_\lambda$ of internally λ-presentable $\mathbf{C}$-objects representing all such objects up to isomorphism.* For all infinite $\kappa < \lambda$, the full subcategory $\mathscr{P}_\kappa \hookrightarrow \mathscr{P}_\lambda$ is given by the internally κ-presentable objects in $\mathscr{P}_\lambda$.

▶ **Definition 4.1.** Let $\kappa \leq \lambda$ be a regular cardinal. A *κ-ary signature* is a set Σ of *operation symbols* σ, each of which is equipped with an *arity* $\mathsf{ar}(\sigma) \in \mathscr{P}_\kappa$.

A *Σ-algebra* A consists of a $\mathbf{C}$-object, also denoted A, and a family of $\mathbf{C}$-morphisms

$$\sigma_A \colon [\mathsf{ar}(\sigma), A] \to A \qquad (\sigma \in \Sigma)$$

A *homomorphism* from A to a Σ-algebra B is a morphism $h \colon A \to B$ in $\mathbf{C}$ such that the diagram below commutes for all $\sigma \in \Sigma$.

$$\begin{array}{ccc} [\mathsf{ar}(\sigma), A] & \xrightarrow{\sigma_A} & A \\ {\scriptstyle h\cdot(-)}\downarrow & & \downarrow{\scriptstyle h} \\ [\mathsf{ar}(\sigma), B] & \xrightarrow{\sigma_B} & B \end{array}$$

We write $\mathsf{Alg}\,\Sigma$ for the category of Σ-algebras and homomorphisms. By a *subalgebra* of the Σ-algebra A, we understand a Σ-algebra B equipped with a homomorphism $h \colon B \hookrightarrow A$ whose underlying $\mathbf{C}$-morphism is an embedding.

Signatures and their algebras. *Fix a κ-ary signature Σ for the remainder of this section.* The category $\mathsf{Alg}\,\Sigma$ can be presented as a category of functor algebras:

▶ **Definition 4.2.** The *signature functor associated to* Σ, $H_\Sigma \colon \mathbf{C} \to \mathbf{C}$, is given by

$$H_\Sigma = \coprod_{\sigma\in\Sigma} [\mathsf{ar}(\sigma), -].$$

The categories $\mathsf{Alg}\,\Sigma$ and $\mathsf{Alg}\,H_\Sigma$ are clearly isomorphic as concrete categories over $\mathbf{C}$, so the forgetful functor $\mathsf{Alg}\,\Sigma \to \mathbf{C}_0$ inherits all properties of the forgetful functor $\mathsf{Alg}\,H_\Sigma \to \mathbf{C}_0$. We collect a few basic consequences of this observation:

▶ **Remark 4.3.**

1. In general, the forgetful functor $\mathcal{U} \colon \mathsf{Alg}\,F \to \mathscr{C}$ from the category $\mathsf{Alg}\,F$ of F-coalgebras for a functor F on a category $\mathscr{C}$ creates all limits in $\mathscr{C}$. It follows that $\mathsf{Alg}\,\Sigma$ has all limits, and the forgetful functor $\mathsf{Alg}\,\Sigma \to \mathbf{C}_0$ creates them.
2. Since H_Σ is a colimit of κ-accessible functors $[\mathsf{ar}(\sigma), -]$, it is itself κ-accessible, so that the forgetful functor $\mathsf{Alg}\,H_\Sigma \to \mathbf{C}_0$ creates κ-directed colimits, and the same holds for the forgetful functor $\mathsf{Alg}\,\Sigma \to \mathbf{C}_0$.
3. From the previous observation (which implies that H_Σ is also λ-accessible) and Proposition 3.8, we obtain by [4, Remark 2.75] (for λ-accessible functors F on locally λ-presentable categories, $\mathsf{Alg}\,F$ is locally λ-presentable) that $\mathsf{Alg}\,\Sigma$ *is locally λ-presentable.*
4. Adámek [1] shows that for a λ-accessible functor F on a cocomplete category $\mathscr{C}$, the forgetful functor $\mathsf{Alg}\,F \to \mathscr{C}$ is right adjoint. From 2 and cocompleteness of $\mathbf{C}_0$ (Section 3), we thus obtain that the forgetful functor $\mathsf{Alg}\,\Sigma \to \mathbf{C}_0$ is right adjoint; that is, *every object $X \in \mathbf{C}$ generates a free Σ-algebra $F_\Sigma X$.*

Varieties of Σ-Algebras. We now describe a syntax for specifying full subcategories of $\mathsf{Alg}\,\Sigma$. As a first step, we introduce a notion of Σ-term, defined as usual in universal algebra:

▶ **Definition 4.4** (Σ-Terms; substitution)**.** For $X \in \mathbf{C}$, we call its underlying set $|X|$ the set of *variables* in X. The set $T_\Sigma(X)$ of *Σ-terms in X* is defined inductively as follows:

1. Each variable in $|X|$ is a Σ-term in X;
2. For each $\sigma \in \Sigma$ and each map $f \colon |\mathsf{ar}(\sigma)| \to T_\Sigma(X)$, $\sigma(f)$ is a Σ-term in X.

We usually omit the signature Σ from the notation and speak simply of *terms* (in X). We employ standard syntactic notions: A *substitution* is a map $\tau\colon |Y| \to T_\Sigma(X)$, for $X, Y \in \mathbf{C}$. We extend τ to a map $\bar{\tau}$ on terms $t \in T_\Sigma(X)$ as usual. Formally, we define $\bar{\tau}(t)$ inductively by $\bar{\tau}(x) = \tau(x)$ for $x \in X$, and $\bar{\tau}\sigma(f) = \sigma(\bar{\tau} \cdot f)$ for $f\colon \mathsf{ar}(\sigma) \to T_\Sigma(X)$. We will not further distinguish between τ and $\bar{\tau}$ in the notation, writing $\tau(t) = \bar{\tau}(t)$ and $\tau \cdot f = \bar{\tau} \cdot f$ for t, f as above. Moreover, the set $\mathsf{sub}(t)$ of *subterms* of a term $t \in T_\Sigma(X)$ is defined as usual; formally, we simultaneously define $\mathsf{sub}(t)$ and $\mathsf{sub}(f)$ for $f\colon I \to T_\Sigma(X)$ (with I some index set or object) inductively by $\mathsf{sub}(x) = \{x\}$ for $x \in X$; $\mathsf{sub}(\sigma(f)) = \{\sigma(f)\} \cup \mathsf{sub}(f)$ for $f\colon \mathsf{ar}(\sigma) \to T_\Sigma(X)$; and $\mathsf{sub}(f) = \bigcup_{i \in I} \mathsf{sub}(f(i))$.

Note that term formation operates without regard for the relational structure. Consequently, the evaluation of terms in a given Σ-algebra may fail to be defined:

▶ **Definition 4.5.** Let A be a Σ-algebra. For an object $X \in \mathbf{C}$ and a relation-preserving assignment $e\colon X \to A$, the partial *evaluation map* $e^\#\colon T_\Sigma(X) \to A$ is inductively defined by

1. $e^\#(x) = e(x)$ for $x \in X$, and
2. $e^\#(\sigma(f))$ is defined for $\sigma \in \Sigma$ and $f\colon |\mathsf{ar}(\sigma)| \to T_\Sigma(X)$ iff the following hold:
   a. $e^\#(f(i))$ is defined for all $i \in \mathsf{ar}(\sigma)$, and
   b. if $\alpha(g)$ is a Π-edge in $\mathsf{ar}(\sigma)$, then $A \models \alpha(e^\# \cdot (f \cdot g))$.

   In case $e^\#(\sigma(f))$ is defined, we put $e^\#(\sigma(f)) = \sigma_A(e^\# \cdot f)$.

As indicated previously, we phrase theories using the relations in Π:

▶ **Definition 4.6.** A Σ-*relation* $X \vdash \alpha(f)$ consists of a *context* $X \in \mathbf{C}$ and a Π-edge $\alpha(f)$ in $T_\Sigma(X)$. We say that $X \vdash \alpha(f)$ is κ-*ary* if $X \in \mathscr{P}_\kappa$. A Σ-algebra A *satisfies* $X \vdash \alpha(f)$ if, for each relation preserving assignment $e\colon X \to A$, $e^\# \cdot f(i)$ is defined for all $i \in X$, and $\alpha_A(e^\# \cdot f)$. A (κ-*ary*) *relational algebraic theory* $(\Sigma, \mathcal{E})$ consists of the (κ-ary) signature Σ and a set $\mathcal{E}$ of κ-ary Σ-relations. It determines the subcategory $\mathsf{Alg}(\Sigma, \mathcal{E})$ of $\mathsf{Alg}\,\Sigma$ consisting of those Σ-algebras which satisfy each Σ-relation in $\mathcal{E}$. We refer to categories of the shape $\mathsf{Alg}(\Sigma, \mathcal{E})$ as *varieties* of Σ-algebras.

▶ **Remark 4.7.** For $\mathbf{C}_0 = \mathbf{Pos}$, the above notion of variety of Σ-algebras corresponds precisely to what we have termed "varieties of coherent algebras" in earlier work with Adámek [5].

▶ **Example 4.8.** Recall that a (1-bounded) metric space X is *complete* if every Cauchy sequence $(x_i)_{i \in \omega}$ of points in X has a limit in X. That is, if (x_i) satisfies the Cauchy property

$$\forall \epsilon > 0.\ \exists N_\epsilon \in \omega.\ \forall n, m \geq N_\epsilon\ (d(y_n, y_m) < \epsilon), \tag{4.1}$$

then there is a point $\mathsf{lim}(x_i) \in X$ with the property of a limit: for all $\epsilon > 0$ there is $N \in \omega$ such that $x_n =_\epsilon \mathsf{lim}(x_i)$ for all $n \geq N$. The full subcategory $\mathbf{CMS} \hookrightarrow \mathbf{Met}$ of complete metric spaces is specified by the relational algebraic theory described below. Thus, by Theorem 4.13 below, we recover the fact that $\mathbf{CMS}$ is monadic over $\mathbf{Met}$. Furthermore, we obtain a completely syntactic ω_1-ary description of the metric completion monad via the deduction system introduced later in this section.

The *theory* $\mathbb{T}_{\mathbf{CMS}}$ *of complete metric spaces* has Γ-ary *limit operations* lim_Γ for all spaces $\Gamma \in \mathscr{P}_{\omega_1}$ of the form $\Gamma = \{x_i \mid i \in \omega\}$ where $(x_i)_{i \in \omega}$ is a Cauchy sequence in Γ. The axioms of $\mathbb{T}_{\mathbf{CMS}}$ then say precisely that $\mathsf{lim}_\Gamma(x_i)$ is a limit of (x_i). Explicitly, for all Γ as above, we impose all axioms of the shape

$$\Gamma \vdash \mathsf{lim}_\Gamma(x_n) =_\epsilon x_k \quad (k \geq N_\epsilon) \qquad \text{where } N_\epsilon \text{ is as in (4.1).}$$

We fix a variety $\mathcal{V} = \mathsf{Alg}(\Sigma, \mathcal{E})$ *for the remainder of this section.* We are going to see that $\mathcal{V}$ is a reflective subcategory of $\mathsf{Alg}\,\Sigma$ by application of Theorem 2.2, i.e. we show that $\mathcal{V}$ is closed under limits and κ-directed colimits in $\mathsf{Alg}\,\Sigma$. We state the second property separately:

▶ **Proposition 4.9.** *$\mathcal{V}$ is closed under κ-directed colimits in $\mathsf{Alg}\,\Sigma$.*

Combining this with Remark 4.3.2, we obtain

▶ **Corollary 4.10.** *The forgetful functor $V\colon \mathcal{V} \to \mathbf{C}$ is κ-accessible.*

It is fairly straightforward to show that $\mathcal{V}$ is also closed under products and subobjects, and hence under limits (in $\mathsf{Alg}\,\Sigma$). Thus, as announced, we have:

▶ **Proposition 4.11.** *$\mathcal{V}$ is a reflective subcategory of $\mathsf{Alg}\,\Sigma$.*

▶ **Remark 4.12.** It follows that the forgetful functor $\mathcal{V} \to \mathbf{C}_0$ has a left adjoint, namely the composite $\mathbf{C} \xrightarrow{F_\Sigma} \mathsf{Alg}\,\Sigma \xrightarrow{R_\mathcal{V}} \mathcal{V}$, where F_Σ is the left adjoint of the forgetful functor $\mathsf{Alg}\,\Sigma \to \mathbf{C}$ (Remark 4.3.4) and $R_\mathcal{V}$ is the reflector according to Proposition 4.11. We call the ensuing monad $\mathbb{T}_\mathcal{V}$ the *free-algebra monad* of $\mathcal{V}$; by Corollary 4.10, $\mathbb{T}_\mathcal{V}$ is κ-accessible.

Indeed, $\mathcal{V}$ is essentially the category of Eilenberg-Moore algebras of $\mathbb{T}_\mathcal{V}$: Using Beck's monadicity theorem, one can show that

▶ **Theorem 4.13.** *The forgetful functor $\mathcal{V} \to \mathbf{C}_0$ is monadic.*

▶ **Corollary 4.14.** *Every κ-ary relational algebraic theory may be translated into an enriched κ-accessible monad, preserving categories of models.*

Relational Logic. We proceed to set up a system of rules for deriving relations among terms. The calculus will involve two forms of judgements, both mentioning a context $X \in \mathbf{Str}(\Pi)$ (not necessarily κ-presentable). By a *relational judgement*

$$X \vdash \alpha(t_1, \dots, t_{\mathsf{ar}(\alpha)}),$$

where $t_1, \dots, t_{\mathsf{ar}(\alpha)} \in T_\Sigma(X)$, we indicate that for every valuation of X that is *admissible*, i.e. satisfies the relational constraints specified in X, the terms t_i are defined, and the resulting tuple of values is in relation α. We treat expressions $\alpha(t_1, \dots, t_{\mathsf{ar}(\alpha)})$ notationally as edges over $T_\Sigma(X)$, in particular sometimes write them in the form $\alpha(f)$ for $f\colon \mathsf{ar}(\alpha) \to T_\Sigma(X)$. Moreover, a *definedness judgement* of the form

$$X \vdash {\downarrow}t$$

states that t is defined for all admissible valuations of X. (We could encode ${\downarrow}t$ as $\phi(t,t)$ for any $\phi \in Eq(x,y)$ but for technical reasons we prefer to keep definedness judgement distinct from relational judgements.)

The rules of the arising system of *relational logic* are shown below:

$$(\mathsf{Var})\ \frac{}{X \vdash {\downarrow}x}\ (x \in X) \qquad (\mathsf{Ctx})\ \frac{}{X \vdash \alpha(x_1, \dots, x_{\mathsf{ar}(\alpha)})}\ (X \models \alpha(x_1, \dots, x_{\mathsf{ar}(\alpha)}))$$

$$(\mathsf{Mor})\ \frac{\{X \vdash \alpha(f_i(j)) \mid j \in \mathsf{ar}(\sigma)\} \cup \{X \vdash {\downarrow}\sigma(f_i) \mid i \in \mathsf{ar}(\alpha)\}}{X \vdash \alpha(\sigma(f_i))}\ ((f_i\colon \mathsf{ar}(\sigma) \to T_\Sigma(X))_{i \in \mathsf{ar}(\alpha)})$$

$$(\mathsf{E\text{-}Ar})\ \frac{\{X \vdash \alpha(f \cdot g) \mid \alpha(g) \in \mathsf{ar}(\sigma)\} \cup \{X \vdash {\downarrow}f(i) \mid i \in \mathsf{ar}(\sigma)\}}{X \vdash {\downarrow}\sigma(f)}\ (f\colon \mathsf{ar}(\sigma) \to T_\Sigma(X))$$

$$\text{(I-Ar)}\ \frac{\{X \vdash \alpha(\tau \cdot f) \mid \alpha(f) \in \Delta\} \cup \{X \vdash \downarrow\tau(y) \mid y \in \Delta\}}{X \vdash \beta(c)}\ (+)$$

$$\text{(RelAx)}\ \frac{\{X \vdash \tau \cdot \varphi \mid \varphi \in \Phi\} \cup \{X \vdash \downarrow\tau(f(i)) \mid i \in \mathsf{ar}(\alpha)\}}{X \vdash \alpha(\tau \cdot f)}\ \begin{array}{l}(\Phi \implies \alpha(f) \in \mathcal{A},\\ \tau\colon \mathsf{Var} \to T_\Sigma(X))\end{array}$$

$$\text{(Ax)}\ \frac{\{X \vdash \alpha(\tau \cdot f) \mid \alpha(f) \in \Delta\} \cup \{X \vdash \downarrow\tau(y) \mid y \in \Delta\}}{X \vdash \beta(\tau \cdot g)}\ (\Delta \vdash \beta(g) \in \mathcal{E})$$

Recall that both the arities of operations in Σ and the contexts of the κ-ary Σ-relations in $\mathcal{E}$ are in $\mathscr{P}_\kappa$. We assume such a $\Delta \in \mathscr{P}_\kappa$ to be specified as $\Delta = R(Y, E)$ by a κ-presentable object $(Y, E) \in \mathbf{Str}(\Pi)$ (cf. Lemma 3.7, Proposition 3.9, Lemma 3.19, Proposition 3.21); by writing $\phi \in \Delta$ for an edge ϕ, we indicate that $\phi \in E$ (rather than just $\phi \in \mathsf{E}(\Delta)$). The rules (E-Ar) and (I-Ar) apply to every $\sigma \in \Sigma$, and rule (Mor) applies to every $\sigma \in \Sigma$ and every $\alpha \in \Pi$. The side condition $(+)$ of (I-Ar) is the following: for some axiom $\Delta \vdash \gamma(g)$ of $\mathcal{V}$ there is $\sigma(h) \in \mathsf{sub}(g)$, where $h\colon \mathsf{ar}(\sigma) \to T_\Sigma(\Delta)$, such that $\mathsf{ar}(\sigma) \models \beta(k)$ and

$$c = \mathsf{ar}(\beta) \xrightarrow{k} \mathsf{ar}(\sigma) \xrightarrow{h} T_\Sigma(\Delta) \xrightarrow{\tau} T_\Sigma(X).$$

Rule (Mor) captures the fact that operations σ are interpreted as morphisms of type $[\mathsf{ar}(\sigma), A] \to A$, a condition that relates to enrichment of the induced monad. Rule (E-Ar) states that operations are defined when all the constraints given by their arity are satisfied. Rules (RelAx) and (Ax) allow application of the axioms of the Horn theory and the variety, respectively, in both cases instantiated with a substitution. A general substitution rule is not included but admissible. Rule (I-Ar) captures that every axiom of the variety is understood as implying that (under the constraints of the context) all subterms occurring in it are defined, in the sense that the constraints in the arities of the relevant operations hold.

▶ **Remark 4.15.** Instantiating the above system of rules to the theory of partial orders yields essentially the ungraded version of our previous deduction system for graded monads on **Pos** [9], up to the above-mentioned coding of definedness judgements. At first glance, the instantiation to the theory of metric spaces appears to yield a system that differs in several respects from the existing system of quantitative algebra [20]; besides the mentioned absence of a general substitution rule, this concerns most prominently the absence of a cut rule (included in [20]) in our system. These distinctions are only superficial: as mentioned above, the more general substitution rule is admissible in our system, and it follows from completeness (Theorem 4.19) that the cut rule is admissible as well.

▶ **Lemma 4.16.** *The following rules are admissible:*

$$\text{(Arity)}\ \frac{X \vdash \downarrow\sigma(m)}{X \vdash \alpha(m \cdot f)}\ \begin{array}{l}(\mathsf{ar}(\sigma) \models \alpha(f),\\ m\colon |\mathsf{ar}(\sigma)| \to T_\Sigma(X))\end{array} \qquad \text{(Subterm)}\ \frac{X \vdash \alpha(f)}{X \vdash \downarrow u}\ (u \in \mathsf{sub}(f))$$

Constructing free algebras. We now show that relational logic gives rise to a syntactic construction of free algebras in the variety $\mathcal{V}$.

The set $\mathscr{T}_\mathcal{V}(X)$ of *derivably $\mathcal{V}$-defined terms in X* consists of those terms $t \in T_\Sigma(X)$ such that $X \vdash \downarrow t$ is derivable. We equip $\mathscr{T}_\mathcal{V}(X)$ with the relations

$$\mathscr{T}_\mathcal{V}(X) \models \alpha(f) \iff X \vdash \alpha(f) \text{ is derivable} \qquad (\alpha \in \Pi, f\colon \mathsf{ar}(\alpha) \to \mathscr{T}_\mathcal{V}(X))$$

making it into a Π-structure. We write $\sim$ for the relation on $\mathscr{T}_\mathcal{V}(X)$ given by *derivable equality*: that is, for all $s, t \in \mathscr{T}_\mathcal{V}(\Gamma)$ we put $s \sim t$ iff $X \vdash \varphi$ is derivable for all $\varphi \in Eq(s, t)$, which is clearly an equivalence relation. The $\sim$-equivalence class of $t \in \mathscr{T}_\mathcal{V}(X)$ is denoted by $[t]$. We

pick a *splitting* $u\colon \mathscr{T}_{\mathcal{V}}(X)/{\sim} \to \mathscr{T}_{\mathcal{V}}(X)$ of the canonical quotient map $q\colon \mathscr{T}_{\mathcal{V}}(X) \to \mathscr{T}_{\mathcal{V}}(X)/{\sim}$, i.e. $q \cdot u = \mathsf{id}$, so u picks representatives of $\sim$-equivalence classes. Then $\mathscr{T}_{\mathcal{V}}(X)/{\sim}$ carries the structure of a $\mathbf{C}$-object, with edges defined by $\mathscr{T}_{\mathcal{V}}(X)/{\sim} \models \alpha(f)$ iff $\mathscr{T}_{\mathcal{V}}(X) \models \alpha(u \cdot f)$. ("Only if" means that u is relation preserving.)

▶ **Definition 4.17.** The *algebra $\mathscr{F}X$ of defined terms in X* is the Σ-algebra obtained by equipping $\mathscr{T}_{\mathcal{V}}(X)/{\sim}$ with the operations $\sigma_{\mathscr{F}X}\colon [\mathsf{ar}(\sigma), \mathscr{T}_{\mathcal{V}}(X)/{\sim}] \to \mathscr{T}_{\mathcal{V}}(X)/{\sim}$ well-defined by $f \mapsto [\sigma(u \cdot f)]$, where $u\colon \mathscr{F}\Gamma \to \mathscr{T}_{\mathcal{V}}(X)$ is the chosen splitting of $q\colon \mathscr{T}_{\mathcal{V}}(X) \to \mathscr{F}X$.

▶ **Theorem 4.18.** *For every $X \in \mathbf{C}$, $\mathscr{F}X$ is a free algebra in $\mathcal{V}$.*

Thus, we see that the free-algebra monad $\mathbb{T}_{\mathcal{V}}$ (Remark 4.12) of a variety $\mathcal{V}$ maps each $X \in \mathbf{Str}(\mathscr{H})$ to the carrier of the algebra $\mathscr{F}X$. We note that the reflection $\mathsf{Alg}\,\Sigma \to \mathcal{V}$ (see Proposition 4.11) need not be epi: the rule (Ax) generally "adds" new defined terms in the presence of axioms; see Adámek et al. [5, Ex. 3.25] for a more detailed view on this point.

▶ **Theorem 4.19** (Soundness and Completeness)**.** *$X \vdash \alpha(f)$ is derivable iff every $A \in \mathcal{V}$ satisfies $X \vdash \alpha(f)$.*

▶ **Remark 4.20.** Consequently, our system instantiated to the theory of metric spaces and the system of quantitative algebra [20], which is also sound and complete, are deductively equivalent. Hence, our results thus far imply that every quantitative algebraic theory induces an ω_1-accessible monad. Indeed this remains true if one admits operations of countable arity, as in our theory of complete metric spaces (Example 4.8). Due to non-discrete contexts in axioms, monads induced by quantitative algebraic theories (such as $x =_{1/2} y \vdash x =_0 y$) in general fail to be finitary. However, our results do imply that the induced monad is finitary if only discrete contexts are used; e.g. this holds for the theories of left-invariant barycentric algebras and of quantitative semi-lattices, respectively [20] (note for the latter that axiom (S4) can be omitted in [20, Def. 9.1]). We conjecture that monads induced by *continuous equation schemes* [20] are also finitary.

5 Enriched Accessible Monads

We proceed to establish the monad-to-theory direction of our correspondence; as already indicated, given our fixed λ-ary Horn theory $\mathscr{H}$, this works only for λ-accessible monads and λ-ary theories, but not for accessibility degrees $\kappa < \lambda$ as in the theory-to-monad direction. So let $\mathbb{T} = (T, \eta, \mu)$ be an enriched λ-accessible monad on $\mathbf{C}$. We proceed to extract a λ-ary relational algebraic theory from $\mathbb{T}$. We first review the equivalence between monads and *Kleisli triples* (see, e.g., Moggi [21], and originally Manes [19, Exercise 12]).

▶ **Definition 5.1.** A *Kleisli triple* in $\mathbf{C}_0$ is a triple $(T, \eta, (-)^*)$ consisting of a mapping $T\colon \mathbf{C}_0 \to \mathbf{C}_0$ (of objects), a morphism $\eta_X\colon X \to TX$ for all $X \in \mathbf{C}_0$, and an assignment of a morphism $f^*\colon TX \to TY$ to every morphism $f\colon X \to TY$. This data is subject to the laws below for all $X \in \mathbf{C}_0$ and all morphisms $f\colon X \to TY$ and $g\colon Y \to TZ$:

$$\eta_X^* = \mathsf{id}_X, \qquad f^* \cdot \eta_X = f, \qquad \text{and} \qquad g^* \cdot f^* = (g^* \cdot f)^*. \tag{5.1}$$

▶ **Remark 5.2.** The mapping which assigns to each monad (T, η, μ) the Kleisli triple $(T, \eta, (-)^*)$ with $(-)^*$ defined by $f^* = TX \xrightarrow{Tf} TTY \xrightarrow{\mu_Y} TY$ for $f \in \mathbf{C}_0(X, TY)$ yields a bijective correspondence between monads and Kleisli triples on $\mathbf{C}_0$.

▶ **Notation 5.3.** For each operation σ in a signature Σ, we have a term $\sigma(u_{\mathsf{ar}(\sigma)})$, where $u_{\mathsf{ar}(\sigma)}$ is the inclusion $\mathsf{ar}(\sigma) \hookrightarrow T_\Sigma(\mathsf{ar}(\sigma))$. By abuse of notation, we also write σ for $\sigma(u_{\mathsf{ar}(\sigma)})$.

▶ **Definition 5.4.** The λ-ary signature $\Sigma_{\mathbb{T}}$ *induced by* $\mathbb{T}$ is the disjoint union of the sets $|T\Gamma|$ ($\Gamma \in \mathscr{P}_\lambda$), where elements of $|T\Gamma|$ have arity Γ. The *variety* $\mathcal{V}_{\mathbb{T}}$ *induced by* $\mathbb{T}$ is $\mathcal{V}_{\mathbb{T}} = \mathsf{Alg}(\Sigma_{\mathbb{T}}, \mathcal{E}_{\mathbb{T}})$ where $\mathcal{E}_{\mathbb{T}}$ contains all axioms of the following shape, with $\Gamma \in \mathscr{P}_\lambda$:

1. $\Gamma \vdash \alpha(\sigma_1, \dots, \sigma_{\mathsf{ar}(\alpha)})$ for all $\sigma_i \in T\Gamma$ such that $T\Gamma \models \alpha(\sigma_1, \dots, \sigma_{\mathsf{ar}(\alpha)})$
2. $\Gamma \vdash f^*(\sigma) = \sigma(f)$ for all $\Delta \in \mathscr{P}_\lambda$, morphisms $f\colon \Delta \to T\Gamma$, and $\sigma \in |T\Delta|$.
3. $\Gamma \vdash \eta_\Gamma(x) = x$ for every $x \in \Gamma$.

Note that in the second item above, for every $x \in \Delta$ the operation symbol $f(x) \in |T\Gamma|$ is considered as a term according to Notation 5.3. Hence $\sigma(f)$ is a term, too.

We now show that $\mathbb{T}$ is the free-algebra monad of its induced variety $\mathcal{V}_{\mathbb{T}}$. For each $X \in \mathbf{C}$, the $\mathbf{C}$-object TX carries a canonical Σ-algebra structure with each operation σ_{TX} being defined by $\sigma_{TX}(f) := f^*(\sigma)$. We call TX the *canonical algebra over* X.

▶ **Lemma 5.5.** *Every canonical algebra lies in* $\mathcal{V}_{\mathbb{T}}$.

▶ **Theorem 5.6.** *Each enriched* λ*-accessible monad* $\mathbb{T}$ *is the free-algebra monad of its induced variety* $\mathcal{V}_{\mathbb{T}}$*, with the free algebra on* X *given by the canonical algebra* TX*.*

▶ **Remark 5.7.** We have thus shown that given a λ-ary Horn theory $\mathscr{H}$, we we can translate λ-accessible monads on $\mathbf{Str}(\mathscr{H})$ back into λ-ary theories, preserving the notion of model. For example, every ω_1-accessible monad on $\mathbf{Met}$ is induced by an ω_1-ary theory, as illustrated in Example 4.8. The situation is more complicated for κ-ary monads where $\kappa < \lambda$. E.g. we can generate a finitary monad on $\mathbf{Met}$ from a single binary operation of type $\{(x, y) \in A^2 \mid d(x, y) < 1/2\} \to A$. This monad is not induced by any theory with operations of internally finitely presentable (i.e. discrete) arity, in particular neither by an ω-ary theory in our framework nor by a quantitative algebraic theory [20].

6 Conclusions

We have introduced the framework of *relational logic* for reasoning about algebraic structure on categories of (finitary) relational structures axiomatized by possibly infinitary Horn theories, such as partial orders or metric spaces. We have proved soundness and completeness of a generic algebraic deduction system, and we have shown that λ-ary relational algebraic theories are in correspondence with λ-accessible enriched monads when the underlying Horn theory is also λ-ary (where "λ-ary" refers to the arity of operations for relational algebraic theories, and to the number of premisses in axioms for Horn theories). Our results allow for a straightforward specification also of infinitary constructions such as metric completion.

The theory-to-monad direction of the above-mentioned correspondence remains true for κ-ary relational algebraic theories and κ-accessible monads on categories of models of λ-ary Horn theories for $\kappa < \lambda$, e.g. when looking at monads and theories on metric spaces. One open end that we leave for future research is to obtain a more complete coverage of this case, which will require a substantial generalization of both the way arities of operations are defined (these can no longer be taken to be objects of the base category) and in the way the axioms of the theory are organized, likely using more topologically-minded approaches.

References

1 Jirí Adámek. Free algebras and automata realizations in the language of categories. *Comment. Math. Univ. Carolin.*, 15(4):589–602, 1974.

2 Jirí Adámek, Matěj Dostál, and Jirí Velebil. A categorical view of varieties of ordered algebras, 2021. `arXiv:2011.13839`.

3 Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories*. Wiley-Interscience, New York, 1990.

4 Jiří Adámek and Jiří Rosický. *Locally Presentable and Accessible Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1994.

5 Jiří Adámek, Chase Ford, Stefan Milius, and Lutz Schröder. Finitary monads on the category of posets, 2020. `arXiv:2011.14796`.

6 Stephen Bloom. Varieties of ordered algebras. *J. Comput. System Sci.*, 2(13):200–212, 1976.

7 Brian Day. A reflection theorem for closed categories. *J. Pure Appl. Algebra*, 2(1):1–11, 1972.

8 Brian Day and Miguel Laplaza. On embedding closed categories. *Bull. Austral. Math. Soc.*, 18(3):357–371, 1978.

9 Chase Ford, Stefan Milius, and Lutz Schröder. Behavioural preorders via graded monads. In *Logic in Computer Science, LICS 2021*. IEEE, 2021. To appear. Preprint available as arXiv e-print 2011.14339.

10 Richard Garner and John Power. An enriched view on the extended finitary monad-Lawvere theory correspondence. *Logical Methods Comput. Sci.*, 14(1):1–23, 2018.

11 Thomas Jech. *Set Theory*. Springer, 2003.

12 Max Kelly. Structures defined by finite limits in the enriched context, I. *Cah. Topol. Géom. Différ. Catég.*, 23(1):3–42, 1982.

13 Max Kelly and Steve Lack. Finite-product-preserving functors, Kan extensions, and strongly-finitary 2-monads. *Appl. Categorical Struct.*, 1(1):85–94, 1993.

14 Max Kelly and John Power. Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *J. Pure Appl. Algebra*, 15(3):163–179, 1993.

15 Alexander Kurz and Jiří Velebil. Quasivarieties and varieties of ordered algebras: regularity and exactness. *Math. Struct. Comput. Sci.*, 27(7):1153–1194, 2017.

16 Fred Linton. Some aspects of equational theories. In *Proc. Conf. on Categorical Algebra at La Jolla*, pages 84–94. Springer, 1966.

17 Fred Linton. An outline of functorial semantics. In B. Eckmann, editor, *Seminar on Triples and Categorical Homology Theory*, volume 80 of *Lecture Notes Math.*, pages 7–52. Springer, 1969.

18 Rory Lucyshyn-Wright. Enriched algebraic theories and monads for a system of arities. *Theory Appl. Categ.*, 31(5):101–137, 2016.

19 Ernest Manes. *Algebraic Theories*. Springer, 1976.

20 Radu Mardare, Prakash Panangaden, and Gordon Plotkin. Quantitative algebraic reasoning. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Logic in Computer Science, LICS 2016*, pages 700–709. ACM, 2016.

21 Eugenio Moggi. Notions of computations and monads. *Inf. Comput.*, 93(1):55–92, 1991.

22 Koki Nishizawa and John Power. Lawvere theories enriched over a general base. *J. Pure Appl. Algebra*, 213(3):377–386, 2009.

23 Erik Palmgren and Steven Vickers. Partial horn logic and cartesian categories. *Ann. Pure Appl. Log.*, 145(3):314–353, 2007.

24 Gordon Plotkin and John Power. Adequacy for algebraic effects. In Furio Honsell and Marino Miculan, editors, *Foundations of Software Science and Computation Structures, FOSSACS 2001*, volume 2030 of *LNCS*, pages 1–24. Springer, 2001.

25 John Power. Enriched lawvere theories. *Theory Appl. Categories*, 6(7):83–93, 1999.

26 Jiří Rosický. Metric monads, 2021. `arXiv:2012.14641`.

Stream Processors and Comodels

Richard Garner
Department of Mathematics and Statistics, Macquarie University, Sydney, Australia

Abstract

In 2009, Ghani, Hancock and Pattinson gave a coalgebraic characterisation of stream processors $A^{\mathbb{N}} \to B^{\mathbb{N}}$ drawing on ideas of Brouwerian constructivism. Their stream processors have an *intensional* character; in this paper, we give a corresponding coalgebraic characterisation of *extensional* stream processors, i.e., the set of continuous functions $A^{\mathbb{N}} \to B^{\mathbb{N}}$. Our account sites both our result and that of *op. cit.* within the apparatus of *comodels* for algebraic effects originating with Power–Shkaravska.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Automata over infinite objects

Keywords and phrases Comodels, residual comodels, bimodels, streams, stream processors

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.15

Funding *Richard Garner*: supported by ARC grants FT160100393 and DP190102432.

1 Introduction

As is well known, the type of infinite *streams* of elements of some type A may be defined to be the final coalgebra $\nu X.\, A \times X$. If types are mere sets, then this coalgebra is manifested as the set $A^{\mathbb{N}}$ of infinite lists of A-elements, with the structure map

$$\alpha \colon \vec{a} \mapsto (a_0, \partial\vec{a}) \qquad \text{where} \qquad \partial(a_0, a_1, a_2, \dots,) = (a_1, a_2, \dots) \ . \tag{1}$$

Of course, the coalgebra structure describes the corecursive nature of streams, but also captures their sequentiality: an A-stream is *first* an A-value, and *then* an A-stream.

If A and B are types, then an *A-B-stream processor* is a way of turning an A-stream into a B-stream. If types are sets, then the crudest kind of stream processor would simply be a function $f \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$; however, it is more computationally reasonable to restrict to those f which are *productive*, in the sense that determining each B-token of the output should require examining only a finite number of A-tokens of the input.

The productive functions $f \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$ are in fact precisely the *continuous* ones for the prodiscrete (= Baire) topologies on $A^{\mathbb{N}}$ and $B^{\mathbb{N}}$. While this representation of stream processors is mathematically smooth, it fails to make explicit their sequentiality: we should like to see the fact that determining each *successive* token of the output B-stream requires examining *successive* finite segments of the input A-stream. Much as for streams themselves, this can be done by presenting stream processors as a final coalgebra.

Such a presentation was given in [7]. Therein, the *type of A-B-stream processors* was taken to be the final coalgebra $\nu X.\, T_A(B \times X)$, where $T_A(V) = \mu X.\, V + X^A$; and it was explained how each element of this type encodes a continuous function $A^{\mathbb{N}} \to B^{\mathbb{N}}$, and how, conversely, each such continuous function yields an element of this type. An interesting aspect of the story is that these assignments are *not* mutually inverse: distinct elements of $\nu X.\, T_A(B \times X)$ may represent the same continuous function, so that elements of this type are really *intensional* representations of stream-processing algorithms.

While there are many perspectives from which this is a good thing, it leaves open the question of whether there is a coalgebraic representation for *extensional* stream processors, i.e., for the bare set of continuous functions $A^{\mathbb{N}} \to B^{\mathbb{N}}$. In this paper, we show that there is:

▶ **Theorem 1.** *Let A and B be sets. The set of continuous functions $A^{\mathbb{N}} \to B^{\mathbb{N}}$ is the underlying set of the terminal B-ary comagma in the category of A-ary magmas.*

In this result, an *A-ary magma* is a set X with an operation $\xi\colon X^A \to X$ satisfying no further axioms. More generally, we can speak of A-ary magmas in any category $\mathcal{C}$ with products; while, if $\mathcal{C}$ is a category with *co*products, we can define an A-ary *co*magma in $\mathcal{C}$ to be an A-ary magma in $\mathcal{C}^{\mathrm{op}}$. Explicitly, this involves an object $X \in \mathcal{C}$ and a map $X \to X + \cdots + X$ into the coproduct of A copies of X, subject to no further conditions.

On the face of it, our Theorem 1 has no obvious relation to [7], nor to anything resembling computation. Thus, the broader contribution of this paper is to site both the ideas of [7] and our Theorem 1 within the well-established machinery of *comodels* [21, 20], as we now explain.

The category-theoretic approach to computational effects originates in [16]: given a monad T on a category of types and programs, we view elements of $T(V)$ as computations with side-effects from T returning values in V. This idea was refined in [19]; rather than considering monads *simpliciter*, we generate them from *algebraic theories* whose basic operations are the computational primitives for the effects at issue. A key example, for us, is the theory $\mathbb{T}_A$ of input from an alphabet A, which is freely generated by a single A-ary operation read.

The approach via algebraic theories has the virtue of giving a good notion of *model* in any category with finite powers. In particular, one has *comodels*, which are models in the opposite of the category of sets, and a key insight of [21] is that comodels of a theory $\mathbb{T}$ can be seen as coalgebraic objects for evaluating $\mathbb{T}$-computations to values: for example, a $\mathbb{T}_A$-comodel is an $A \times (-)$-coalgebra, and the *final* comodel is the set of A-streams.

A range of authors [20, 15, 17, 18, 23, 10, 1, 6, 24, 5] have taken this attractive perspective on operational semantics further. Particularly salient for us is the concept, due to [1, 10, 24] of a *residual comodel*. Given theories $\mathbb{T}$ and $\mathbb{R}$, an $\mathbb{R}$-residual $\mathbb{T}$-comodel is, formally speaking, a comodel of $\mathbb{T}$ in the Kleisli category of $\mathbb{R}$ but is, practically speaking, a coalgebraic entity for evaluating or compiling $\mathbb{T}$-computations into $\mathbb{R}$-computations. In particular, we have $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodels, which translate requests for B-input into requests for A-input, and a little thought shows that this is exactly the rôle filled by an A-B-stream processor. In fact, the final coalgebra of [7] turns out to be precisely the *final $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel*; in §3 we explain this, and show how other aspects of [7] flow naturally from this fact.

To get from here to our Theorem 1 requires a new import from category-theoretic universal algebra: the notion of a *bimodel* [4, 22, 3]. An *$\mathbb{R}$-$\mathbb{T}$-bimodel* is a comodel of $\mathbb{T}$ in the category of *Set*-models of $\mathbb{R}$. Since this latter category has the Kleisli category as a full subcategory, bimodels are a generalisation of residual comodels – one which, roughly speaking, allows additional quotients by bisimulations to be taken. These quotients are just what one needs to collapse the intensional stream processors of [7] to their underlying continuous functions: so yielding our Theorem 1 which we now recognise as describing the *final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel*.

There is plenty more to be said in this direction: our results map a path toward studying residual comodels and bimodels of other theories, with notions like hidden Markov models, or non-deterministic and Rabin automata, all within scope. But this must await another paper!

2 Streams as a final comodel

In this background section, we recall how algebraic theories present notions of effectful computation, how *comodels* of a theory furnish environments appropriate for evaluating such computations, and how the type of streams arises as a final comodel.

▶ **Definition 2** (Algebraic theory). *A* signature *comprises a set* Σ *of* function symbols, *and for each* $\sigma \in \Sigma$ *a set* $|\sigma|$, *its* arity. *Given a signature* Σ *and a set* V, *we define the set* $\Sigma(V)$ *of* Σ-terms with variables in V *by the inductive clauses*

$$v \in V \implies v \in \Sigma(V) \quad \text{and} \quad \sigma \in \Sigma,\, t \in \Sigma(V)^{|\sigma|} \implies \sigma(t) \in \Sigma(V)\ .$$

An equation *over a signature* Σ *is a formal equality* $t = u$ *between terms in the same set of free variables. A (algebraic)* theory $\mathbb{T}$ *comprises a signature and a set of equations over it.*

▶ **Definition 3** ($\mathbb{T}$-terms). *Given a signature* Σ *and terms* $t \in \Sigma(V)$ *and* $u \in \Sigma(W)^V$, *we define the* substitution $t(u) \in \Sigma(W)$ *by recursion on* t*:*

$$v \in V \implies v(u) = u_v \quad \text{and} \quad \sigma \in \Sigma,\, t \in \Sigma(V)^{|\sigma|} \implies (\sigma(t))(u) = \sigma(\lambda i.\, t_i(u))\ .$$

Given a theory $\mathbb{T}$ *with signature* Σ, *we define* $\mathbb{T}$-equivalence *as the smallest family of substitution-congruences* $\equiv_{\mathbb{T}}$ *on the sets* $\Sigma(V)$ *such that* $t \equiv_{\mathbb{T}} u$ *for all equations* $t = u$ *of* $\mathbb{T}$. *The set* $T(V)$ *of* $\mathbb{T}$-terms with variables in V *is* $\Sigma(V)/\equiv_{\mathbb{T}}$.

When a theory $\mathbb{T}$ is seen as specifying a computational effect, $T(V)$ describes the set of computations with effects from $\mathbb{T}$ returning a value in V. There are theories for effects such as output, state, exceptions, and so on, but for us the salient example is:

▶ **Example 4** (Input). Given a set A, the theory $\mathbb{T}_A$ of *A-valued input* comprises a single A-ary function symbol read, satisfying no equations, whose action we think of as:

$$(t\colon A \to X) \mapsto \mathsf{let}\ \mathsf{read}()\ \mathsf{be}\ a.\, t(a)\ .$$

The set of terms $T_A(V)$ is, as in the introduction, the initial algebra $\mu X.\, V + X^A$, whose elements may be seen combinatorially as A-ary branching trees with leaves labelled in V; or computationally as programs which request A-values from an external source and use them to determine a return value in V. For example, when $A = \mathbb{N}$, the program which requests two input values and returns their sum is presented by

$$\mathsf{let}\ \mathsf{read}()\ \mathsf{be}\ n.\, \mathsf{let}\ \mathsf{read}()\ \mathsf{be}\ m.\, n + m\ \in\ T(\mathbb{N})\ . \tag{2}$$

We now define the models of an algebraic theory. In the definition, we say that a category $\mathcal{C}$ has *powers* if it has all set-indexed self-products $X^A := \Pi_{a \in A} X$.

▶ **Definition 5** (Σ-structure, $\mathbb{T}$-model). *Let* Σ *be a signature. A* Σ-structure $\boldsymbol{X}$ *in a category* $\mathcal{C}$ *with powers is an object* $X \in \mathcal{C}$ *with* operations $[\![\sigma]\!]_{\boldsymbol{X}} : X^{|\sigma|} \to X$ *for each* $\sigma \in \Sigma$. *For each* $t \in \Sigma(V)$ *the* derived operation $[\![t]\!]_{\boldsymbol{X}} : X^V \to X$ *is then determined by the recursive clauses:*

$$[\![v]\!]_{\boldsymbol{X}} = \pi_v \qquad \text{and} \qquad [\![\sigma(t)]\!]_{\boldsymbol{X}} = X^V \xrightarrow{([\![t_i]\!]_{\boldsymbol{X}})_{i \in |\sigma|}} X^{|\sigma|} \xrightarrow{[\![\sigma]\!]_{\boldsymbol{X}}} X\ . \tag{3}$$

Given a theory $\mathbb{T} = (\Sigma, \mathcal{E})$, *a* $\mathbb{T}$-model in $\mathcal{C}$ *is a* Σ*-structure* $\boldsymbol{X}$ *which satisfies* $[\![t]\!]_{\boldsymbol{X}} = [\![u]\!]_{\boldsymbol{X}}$ *for all equations* $t = u$ *of* $\mathbb{T}$. *The unqualified term "model" will always mean "model in Set". We write* $\mathit{Mod}(\mathbb{T}, \mathcal{C})$ *for the category of* $\mathbb{T}$*-models in* $\mathcal{C}$, *and* $\mathit{Mod}(\mathbb{T})$ *for the models in Set.*

The set of computations $T(V)$ has a structure of $\mathbb{T}$-model $\boldsymbol{T}(V)$ with operations given by substitution; and as is well known, this structure is in fact *free*:

▶ **Lemma 6.** *The inclusion of variables* $\eta_V\colon V \to T(V)$ *exhibits* $\boldsymbol{T}(V)$ *as the free* $\mathbb{T}$*-model on* V. *That is, for any* $\mathbb{T}$*-model* $\boldsymbol{X}$ *and any function* $f\colon V \to X$, *there is a unique* $\mathbb{T}$*-model homomorphism* $f^\dagger\colon \boldsymbol{T}(V) \to \boldsymbol{X}$ *with* $f^\dagger \circ \eta_V = f$. *Explicitly,* $f^\dagger(t) = [\![t]\!]_{\boldsymbol{X}}(\lambda v.\, f(v))$.

Taking the full subcategory of $Mod(\mathbb{T})$ on the free $\mathbb{T}$-models yields the well known *Kleisli category* of $\mathbb{T}$, which we typically present as follows:

▶ **Definition 7** (Kleisli category). *The* Kleisli category $Kl(\mathbb{T})$ *of a theory* $\mathbb{T}$ *has sets as objects; hom-sets* $Kl(\mathbb{T})(A,B) = Set(A,TB)$*; the identity at* A *being* $\eta_A \colon A \to TA$*; and composition* $g, f \mapsto g^\dagger \circ f$ *with* $g^\dagger$ *as in Lemma 6 for the free* $\mathbb{T}$*-model structures. The* free functor $F_{\mathbb{T}} \colon Set \to Kl(\mathbb{T})$ *is the identity on objects and sends* $f \in Set(X,Y)$ *to* $\eta_Y \circ f \in Kl(\mathbb{T})(X,Y)$*. The fully faithful* comparison functor $I_{\mathbb{T}} \colon Kl(\mathbb{T}) \to Mod(\mathbb{T})$ *maps* $A \mapsto \boldsymbol{T}A$ *and* $f \mapsto f^\dagger$*.*

The Kleisli category captures the compositionality of computations with effects from $\mathbb{T}$, and allows us to draw the link with Moggi's monadic semantics [16]; indeed, the free functor $F_{\mathbb{T}} \colon Set \to Kl(\mathbb{T})$ and its right adjoint $Kl(\mathbb{T})(1,-) \colon Kl(\mathbb{T}) \to Set$ generate an associated monad T on Set and we have that $Kl(\mathbb{T}) \cong Kl(\mathsf{T})$ under Set.

So far we have said nothing about *non*-free $\mathbb{T}$-models. It is a basic fact that every such model can be obtained from a free one by quotienting by some congruence, and so can been seen as a set of computations identified up to some notion of program equivalence. This is important, for example, in [12], and will be important for us in §4 below.

We now turn from models to the dual notion of *comodel*. We say a category $\mathcal{C}$ has *copowers* if if each set-indexed self-coproduct $A \cdot X = \Sigma_{a \in A} X$ exists in $\mathcal{C}$.

▶ **Definition 8** ($\mathbb{T}$-comodel). *Let* $\mathbb{T}$ *be a theory. A* $\mathbb{T}$-comodel *in a category* $\mathcal{C}$ *with copowers is a model of* $\mathbb{T}$ *in* $\mathcal{C}^{\mathrm{op}}$*, comprising an object* $S \in \mathcal{C}$ *and* co-operations $[\![\sigma]\!]^{\boldsymbol{S}} : S \to |\sigma| \cdot S$ *obeying the equations of* $\mathbb{T}$*. The unqualified term "comodel" will mean "comodel in Set". We write* $Comod(\mathbb{T}, \mathcal{C})$ *for the category of* $\mathbb{T}$*-comodels in* $\mathcal{C}$*, and* $Comod(\mathbb{T})$ *for the comodels in Set.*

As explained in [21, 20], when a theory $\mathbb{T}$ presents a notion of computation, its comodels provide deterministic environments for evaluating computations with effects from $\mathbb{T}$.

▶ **Example 9.** A comodel $\boldsymbol{S}$ of the theory of A-valued input is a state machine that answers requests for A-characters; it comprises a set of states S and a map $[\![\mathsf{read}]\!]^{\boldsymbol{S}} = (g,n) \colon S \to A \times S$ giving for each $s \in S$ a next character $g(s) \in A$ and a next state $n(s) \in S$.

While the comodels of the preceding example are just $A \times (-)$-coalgebras, the comodel perspective adds something to this. The general picture is that a $\mathbb{T}$-comodel allows us to evaluate $\mathbb{T}$-computations $t \in T(V)$ down to values in V via the derived operations of Definition 5. Indeed, given a comodel $\boldsymbol{S}$ and a term $t \in T(V)$, we have the derived co-operation $[\![t]\!]^{\boldsymbol{S}} : S \to V \times S$ which, unfolding the definition, is given by the clauses:

$$\begin{aligned} v \in V &\implies [\![v]\!]^{\boldsymbol{S}}(s) = (v,s) \\ \text{and } \sigma \in \Sigma, t \in T(V)^{|\sigma|} &\implies [\![\sigma(t)]\!]^{\boldsymbol{S}}(s) = [\![t_v]\!]^{\boldsymbol{S}}(s') \text{ where } [\![\sigma]\!]^{\boldsymbol{S}}(s) = (v,s') \ . \end{aligned} \tag{4}$$

The idea is that applying $[\![t]\!]^{\boldsymbol{S}}$ to a starting state $s \in S$ will yield the value $v \in V$ and final state $s' \in S$ obtained by running the computation $t \in T(V)$, using the co-operations of the comodel $\boldsymbol{S}$ to answer the requests posed by the corresponding operation symbols of $\mathbb{T}$.

▶ **Example 10.** For a comodel $(g,n) \colon S \to A \times S$ of A-valued input, the clauses (4) become

$$v \in V \implies [\![v]\!]^{\boldsymbol{S}}(s) = (v,s) \qquad t \in T(V)^A \implies [\![\mathsf{read}(t)]\!]^{\boldsymbol{S}}(s) = [\![t(g(s))]\!]^{\boldsymbol{S}}(n(s)) \ .$$

So if we consider $A = \mathbb{N}$, the term $t = \mathsf{read}(\lambda n.\, \mathsf{read}(\lambda m.\, n+m)) \in T(\mathbb{N})$ from (2), and the comodel $\boldsymbol{S}$ with $S = \{s, s', s''\}$ and $[\![\mathsf{read}]\!]^{\boldsymbol{S}} = (g,n) \colon S \to \mathbb{N} \times S$ given by the upper line in:

$$\begin{aligned} [\![\mathsf{read}]\!]^{\boldsymbol{S}} : &\quad s \mapsto (3,s') &\quad s' \mapsto (6,s'') &\quad s'' \mapsto (11,s'') \\ [\![t]\!]^{\boldsymbol{S}} : &\quad s \mapsto (9,s'') &\quad s' \mapsto (17,s'') &\quad s'' \mapsto (22,s'') \ , \end{aligned}$$

then $[\![t]\!]^{\boldsymbol{S}} : S \to \mathbb{N} \times S$ is given by the lower line. For example, we calculate that $[\![t]\!](s) = [\![\mathsf{read}(\lambda n.\, \mathsf{read}(\lambda m.\, n+m))]\!](s) = [\![\mathsf{read}(\lambda m.\, 3+m)]\!](s') = [\![3+6]\!](s'') = (9, s'')$.

As is idiomatic, the *final* comodel of a theory describes "observable behaviours" that states of a comodel may possess. To make this precise, we define states $s_1 \in \boldsymbol{S}_1$ and $s_2 \in \boldsymbol{S}_2$ of two $\mathbb{T}$-comodels to be *operationally equivalent* if running any $\mathbb{T}$-computation $t \in T(V)$ starting from the state s_1 of $\boldsymbol{S}_1$ or from the state s_2 of $\boldsymbol{S}_2$ gives the same value; i.e.,

$$\text{if} \qquad [\![t]\!]^{\boldsymbol{S}_1}(s_1) = (v_1, s_1') \quad \text{and} \quad [\![t]\!]^{\boldsymbol{S}_2}(s_2) = (v_2, s_2') \qquad \text{then} \qquad v_1 = v_2 \ .$$

▶ **Lemma 11.** *States $s_1 \in \boldsymbol{S}_1$ and $s_2 \in \boldsymbol{S}_2$ of two $\mathbb{T}$-comodels are operationally equivalent if and only if they become equal under the unique maps $\boldsymbol{S}_1 \to \boldsymbol{F} \leftarrow \boldsymbol{S}_2$ to the final $\mathbb{T}$-comodel.*

Proof. This is [5, Proposition 52]. ◀

So in the spirit of [11, Theorem 4], we may (if we adequately handle the set-theoretic issues) characterise the final $\mathbb{T}$-comodel as the set of all possible states of all possible comodels, modulo operational equivalence. For A-valued input, two states s_1, s_2 of two comodels $(g_i, n_i)\colon S_i \to A \times S_i$ are operationally equivalent if they give the same stream of values:

$$\big(g_1(s_1), g_1(n_1(s_1)), g_1(n_1(n_1(s_1))), \dots\big) = \big(g_2(s_2), g_2(n_2(s_2)), g_2(n_2(n_2(s_2))), \dots\big) \ ,$$

and in this way, we may reconstruct the familiar fact that the final $\mathbb{T}_A$-comodel $\boldsymbol{A}^{\mathbb{N}}$ is the set of A-streams $A^{\mathbb{N}}$ with the structure map (1).

The comodel view also allows us to capture the *topology* on the space of streams. Indeed, any comodel of a theory has a natural prodiscrete topology, whose basic open sets describe those states which are indistinguishable with respect to a finite set of $\mathbb{T}$-computations.

▶ **Definition 12** (Operational topology). *Let $\boldsymbol{S}$ be a $\mathbb{T}$-comodel. The* operational topology *on S is generated by sub-basic open sets*

$$[t \mapsto v] := \{s \in S : [\![t]\!]^{\boldsymbol{S}}(s) = (v, s') \text{ for some } s'\} \qquad \text{for all } t \in T(V) \text{ and } v \in V \ .$$

This definition appears to be new, and investigating its force is beyond the scope of this paper; in particular, we have space only to state the following result, whose proof the reader may find an interesting exercise. It implies easily that $\boldsymbol{A}^{\mathbb{N}}$ is the final *topological* comodel when given the topology obtained from the product of discrete topologies on each copy of A.

▶ **Lemma 13.** *For any theory $\mathbb{T}$, the final $\mathbb{T}$-comodel $\boldsymbol{F}$, when endowed with its operational topology, is the final topological comodel.*

3 Intensional stream processors as a final residual comodel

In this section, we recall a more general kind of comodel considered by, among others, [1, 10, 24], which allows for stateful translations between different notions of computation. We then explain how the intensional stream processors of [7] instantiate this notion, and use this fact to derive a number of other aspects of the theory of [7].

▶ **Definition 14** (Residual comodel). *Let $\mathbb{T}$ and $\mathbb{R}$ be theories. An* $\mathbb{R}$-residual $\mathbb{T}$-comodel *is a comodel of $\mathbb{T}$ in the Kleisli category $Kl(\mathbb{R})$.*

The nomenclature "residual" comes from [10, §5.3], and we will explain the connection to *loc. cit.* in Proposition 25 below. For now, let us spell out in detail what a residual comodel $\boldsymbol{S}$ involves. First, there is an underlying set of states S. Next, we have for each $\sigma \in \Sigma$ a basic co-operation $[\![\sigma]\!]^{\boldsymbol{S}} : S \to R(|\sigma| \times S)$ assigning to each state $s \in S$ an $\mathbb{R}$-computation $[\![\sigma]\!]^{\boldsymbol{S}}(s)$ returning values in $|\sigma| \times S$ – where, as before, we think of these two components as providing a value answering the request posed by σ, and a next state. Now we determine a derived co-interpretation $[\![t]\!]^{\boldsymbol{S}} : S \to R(V \times S)$ for each $t \in T(V)$ by threading the basic co-operations together via monadic binding:

$$\begin{aligned} v \in V \subseteq T(V) &\implies [\![v]\!]^{\boldsymbol{S}}(s) = (v, s) \in V \times S \subseteq R(V \times S) \\ \text{and } \sigma \in \Sigma, t \in T(V)^{|\sigma|} &\implies [\![\sigma(t)]\!]^{\boldsymbol{S}}(s) = [\![\sigma]\!]^{\boldsymbol{S}}(s)\big(\lambda(v, s').\ [\![t_v]\!]^{\boldsymbol{S}}(s')\big)\ , \end{aligned} \tag{5}$$

and the final requirement is that these derived operations should satisfy the equations of $\mathbb{T}$.

Interesting examples of residual comodels are given in [1, 24, 9], but for us the key case is:

▶ **Example 15.** A comodel of the theory $\mathbb{T}_B$ of B-valued input residual on the theory $\mathbb{T}_A$ of A-valued input comprises a set of states S, and a function $\gamma\colon S \to T_A(B \times S)$ assigning to each state $s \in S$ a program which uses some number of A-tokens from an input stream to inform the choice of an output B-token and a new state in S.

It is easy to see how each state s_0 of such a comodel should encode a stream processor $A^{\mathbb{N}} \to B^{\mathbb{N}}$: given an input stream $\vec{a} \in A^{\mathbb{N}}$, we consume some initial segment $a_0, \ldots, a_k$ to answer the requests posed by the program $\gamma(s_0)$, so obtaining an element $b_0 \in B$ and a new state s_1. We now repeat starting from $s_1 \in S$ and the remaining part $\partial^k \vec{a}$ of the input stream, to obtain b_1 and s_2 while consuming $a_{k+1}, \ldots, a_\ell$; and so on coinductively. This description was made mathematically precise in [7, §3.1], but in fact we can obtain it in a principled comodel-theoretic manner via (a special case of) a notion given in [20, Appendix].

▶ **Definition 16** (Tensor of a residual comodel with a comodel)**.** *Let $\mathbb{T}$ and $\mathbb{R}$ be theories. Let $\boldsymbol{S}$ be an $\mathbb{R}$-residual $\mathbb{T}$-comodel, and let $\boldsymbol{M}$ be an $\mathbb{R}$-comodel. The* tensor product $\boldsymbol{S} \cdot \boldsymbol{M}$ *is the $\mathbb{T}$-comodel with underlying set $S \times M$ and co-operations*

$$[\![\sigma]\!]^{\boldsymbol{S}\cdot\boldsymbol{M}} : S \times M \xrightarrow{[\![\sigma]\!]^{\boldsymbol{S}} \times M} R(|\sigma| \times S) \times M \xrightarrow{(t,m) \mapsto [\![t]\!]^{\boldsymbol{M}}(m)} |\sigma| \times S \times M\ . \tag{6}$$

This definition makes intuitive sense: given a state machine for translating $\mathbb{T}$-computations into $\mathbb{R}$-computations, and one for executing $\mathbb{R}$-computations, it threads them together to yield a state machine for executing $\mathbb{T}$-computations. We will make this justification rigorous in Definition 26 below, but for the moment let us simply assume its reasonability and give:

▶ **Definition 17** (Extent)**.** *The* extent *of a $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel $\boldsymbol{S}$ is the function $e\colon S \times A^{\mathbb{N}} \to B^{\mathbb{N}}$ underlying the unique map of $\mathbb{T}_B$-comodels $\boldsymbol{S} \cdot \boldsymbol{A}^{\mathbb{N}} \to \boldsymbol{B}^{\mathbb{N}}$, where here $\boldsymbol{A}^{\mathbb{N}}$ and $\boldsymbol{B}^{\mathbb{N}}$ are endowed with their final comodel structures.*

We now unfold this definition. Firstly, for any term $t \in T_A(V)$, the derived co-operation $[\![t]\!]^{\boldsymbol{A}^{\mathbb{N}}}(\vec{a})\colon A^{\mathbb{N}} \to V \times A^{\mathbb{N}}$ is defined recursively by

$$[\![v]\!]^{\boldsymbol{A}^{\mathbb{N}}}(\vec{a}) = (v, \vec{a}) \qquad \text{and} \qquad [\![\mathsf{read}(t)]\!]^{\boldsymbol{A}^{\mathbb{N}}}(\vec{a}) = [\![t(a_0)]\!]\,(\partial\vec{a})\ . \tag{7}$$

If we view t as an A-ary branching tree with leaves labelled in V, then $[\![t]\!]^{\boldsymbol{A}^{\mathbb{N}}}(\vec{a})$ is the result of walking up the tree from the root, consuming an element of $\vec{a}$ at each interior node to

determine which branch to take, and returning at a leaf the V-value found there along with what remains of $\vec{a}$. Now, in terms of this, the $\mathbb{T}_B$-comodel structure of $\boldsymbol{S} \cdot \boldsymbol{A}^{\mathbb{N}}$ is given by

$$S \times A^{\mathbb{N}} \to B \times S \times A^{\mathbb{N}} \qquad (s, \vec{a}) \mapsto [\![\gamma(s)]\!]^{\boldsymbol{A}^{\mathbb{N}}}(\vec{a}) \ ;$$

that is, by the function taking a state s_0 and stream $\vec{a}$ to the triple $(b_0, s_1, \partial^k \vec{a})$ obtained by walking up k nodes of the tree $\gamma(s_0)$ to the leaf (b_0, s_1). If we view this comodel structure as a pair of maps $g \colon S \times A^{\mathbb{N}} \to B$ and $n \colon S \times A^{\mathbb{N}} \to S \times A^{\mathbb{N}}$, then we can say, finally, that the extent function $e \colon S \times A^{\mathbb{N}} \to B^{\mathbb{N}}$ of $\boldsymbol{S}$ is defined coinductively by:

$$\big(e(s, \vec{a})\big)_0 = g(s, \vec{a}) \qquad \partial\big(e(s, \vec{a})\big) = e\big(n(s, \vec{a})\big) \ .$$

Comparing this construction with that of [7, §3.1], done there with bare hands, we find that they are exactly the same: the derived co-operations $[\![t]\!]$ of (7) are the functions *eat t* of *loc. cit.*, while our extent function e is their function eat_∞.

We have thus shown that each state s of a $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel encodes a function $e(s, -) \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$; but for these functions to be reasonable stream processors, they should be *continuous* for the profinite topologies. While this may be shown with little effort, we may in fact see it without *any* effort via a comodel-theoretic argument. We first need:

▶ **Definition 18** (Tensor of a residual comodel and a topological comodel)**.** *Let $\mathbb{T}$ and $\mathbb{R}$ be theories, let $\boldsymbol{S}$ be an $\mathbb{R}$-residual $\mathbb{T}$-comodel, and $\boldsymbol{M}$ a topological $\mathbb{R}$-comodel. The* tensor product *$\boldsymbol{S} \cdot \boldsymbol{M}$ is the topological $\mathbb{T}$-comodel with underlying space $S \cdot M$ and co-operations* (6)*.*

Once again, the justification for this definition will be given below; assuming it for now, the desired continuity of each $e(s, -)$ is immediate. For indeed, viewing $\boldsymbol{A}^{\mathbb{N}}$ and $\boldsymbol{B}^{\mathbb{N}}$ as final topological comodels, there is a unique map of topological $\mathbb{T}_B$-comodels $\boldsymbol{S} \cdot \boldsymbol{A}^{\mathbb{N}} \to \boldsymbol{B}^{\mathbb{N}}$ which, since forgetting the topology yields back a map of *Set*-comodels, must be the extent function e. Thus, the force of this is that e is continuous as a map $S \cdot A^{\mathbb{N}} \to B^{\mathbb{N}}$ – which is equally to say that each $e(s, -) \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$ is continuous for the profinite topologies.

So far, our argument has been given for an arbitrary $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel $\boldsymbol{S}$; but as in [7], it is natural to consider the final residual comodel in particular. To this end, we should first clarify the correct notion of *morphism* between residual comodels.

▶ **Definition 19** (Map of residual comodels)**.** *Let $\mathbb{T}$ and $\mathbb{R}$ be theories, and let $\boldsymbol{S}$ and $\boldsymbol{U}$ be $\mathbb{R}$-residual $\mathbb{T}$-comodels. A map of residual comodels $\boldsymbol{S} \to \boldsymbol{U}$ is a function $f \colon S \to U$ such that $[\![\sigma]\!]^{\boldsymbol{U}} \circ f = R(|\sigma| \times f) \circ [\![\sigma]\!]^{\boldsymbol{T}}$ for all operations σ in the signature of $\mathbb{T}$.*

▶ Remark 20. Given that an $\mathbb{R}$-residual comodel is a comodel in $Kl(\mathbb{R})$, we might expect a map of residual comodels to be a map in $Kl(\mathbb{R})$, rather than one in *Set*. The reason for our unusual choice is not pure expediency; it has to do with an enrichment of the category of theories in the category of comonads on *Set*, currently being investigated by the authors of [10], and which exploits the general *Sweedler theory* of [2]. Working through the calculations, one finds that for two theories $\mathbb{R}$ and $\mathbb{T}$, the category of coalgebras for the hom-comonad $\langle \mathbb{R}, \mathbb{T} \rangle$ is the category of residual comodels, with precisely the maps indicated in Definition 19.

With this clarification made, we see that, in particular, the category of $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodels is simply the category of $T_A(B \times -)$-coalgebras, and so we have:

▶ **Definition 21** (Intensional stream processors)**.** *The* type of intensional A-B-stream processors *is the final $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel $\boldsymbol{I_{AB}}$, i.e., the final coalgebra $\nu X. T_A(B \times X)$. The* reflection *function is the currying of the topological extent function*

$$\mathsf{reflect} \colon I_{AB} \to Top(A^{\mathbb{N}}, B^{\mathbb{N}}) \qquad s \mapsto e(s, -) \colon A^{\mathbb{N}} \to B^{\mathbb{N}} \ ,$$

where here we write Top for the category of topological spaces and continuous maps.

As well as reflection, [7] also defines a *reification* function $\mathsf{reify}\colon \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}}) \to I_{AB}$ that implements each continuous function by a state of the final comodel, and which satisfies $\mathsf{reflect} \circ \mathsf{reify} = \mathrm{id}$. This means that $\mathsf{reflect}$ is surjective – but crucially, it is *not* injective. To show this, we must first note that, by the usual techniques, the terminal coalgebra I_{AB} may be described as follows: it is the set of all finite or infinite A-ary branching trees, with interior nodes labelled with elements of B^* (i.e., lists of elements of B), with leaves labelled by elements of $B^{\mathbb{N}}$, and where no infinite path of interior nodes is labelled by the empty list.

▶ **Example 22.** Fix an element $b \in B$ and consider the following two $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel structures on $\{*\}$:

$$\text{(i)} \ * \mapsto (b, *) \qquad \text{and} \qquad \text{(ii)} \ * \mapsto \mathsf{read}(\lambda a.\, (b, *)) \ . \tag{8}$$

In both comodels, the unique state $*$ encodes the continuous function $A^{\mathbb{N}} \to B^{\mathbb{N}}$ sending every stream $\vec{a}$ to $(b, b, b, b, \dots)$. However, these states yield different elements of the final comodel I_{AB}: (i) gives the trivial tree t_0 whose root is labelled by $(b, b, b, \dots)$, while (ii) gives the purely infinite A-ary branching tree t_1 with every node labelled by a single b.

Intuitively, the two states of I_{AB} in this example differ in that the first ignores its input stream entirely, and simply outputs b's without cease; while the second frivolously consumes a single A-token before emitting each b. So I_{AB} is a set of *intensional* representations of stream processors. This leads us neatly on to the second part of the paper, where we give a comodel-theoretic presentation of *extensional* stream processors, i.e., the set $\mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$, and an explanation in these terms of where the reification function of [7] comes from.

Before doing this, we resolve some unfinished business by justifying Definitions 16 and 18 above. Our starting point will be an alternative presentation of the notion of comodel due to [23]. In *op. cit.*, Uustalu defines a *runner* for a theory $\mathbb{T}$, with set of states S, to be a monad morphism $\mathsf{T} \to \mathsf{T}_S$ from the associated monad of $\mathbb{T}$ to the state monad $\mathsf{T}_S = (- \times S)^S$. The data for such a runner are functions $T(V) \to (V \times S)^S$ assigning to each $t \in T(V)$ a function $[\![t]\!]^{\boldsymbol{S}} : S \to V \times S$. Recognising these as also being the data of the derived co-operations of a $\mathbb{T}$-comodel structure on S, we should find the main result of [23] reasonable: that $\mathbb{T}$-comodels with underlying set S are in bijection with $\mathbb{T}$-runners with underlying set of states S.

While Uustalu's result is about comodels in Set, it generalises unproblematically. For any object S of a category $\mathcal{C}$ with copowers, we have an adjunction $(-) \cdot S \dashv \mathcal{C}(S, -)\colon \mathcal{C} \to \mathit{Set}$ inducing a monad $\mathsf{T}_S = \mathcal{C}(S, (-) \cdot S)$ on Set; in [15] this is called the *linear-use state monad* associated to S. We now have the following natural extension of Uustalu's result.

▶ **Proposition 23** (cf. [15, Theorem 8.2])**.** *Let $\mathbb{T}$ be an algebraic theory, $\mathcal{C}$ a category with copowers, and $S \in \mathcal{C}$. The following are in bijective correspondence:*

1. *$\mathbb{T}$-comodels $\boldsymbol{S}$ in $\mathcal{C}$ with underlying object S;*
2. *$\mathbb{T}$-runners in $\mathcal{C}$, i.e., monad maps $[\![-]\!]^{\boldsymbol{S}} : \mathsf{T} \to \mathsf{T}_S$ into the linear-use state monad of S;*
3. *Functorial extensions of $(-) \cdot S\colon \mathit{Set} \to \mathcal{C}$ along the free functor into the Kleisli category:*

$$\begin{array}{ccc} \mathit{Set} & \xrightarrow{(-)\cdot S} & \mathcal{C} \\ {\scriptstyle F_{\mathbb{T}}}\downarrow & \nearrow_{(-)\cdot \boldsymbol{S}} & \\ \mathit{Kl}(\mathbb{T}) & & \end{array} \ . \tag{9}$$

▶ Remark 24. Abstractly, this proposition expresses the fact that $\mathit{Kl}(\mathbb{T})$ is the *free category with copowers containing a comodel of* $\mathbb{T}$; this result is originally due to Linton [13].

Proof. As just said, the argument for (1) ⇔ (2) is *mutatis mutandis* that of [23, §3]. For (2) ⇔ (3), it is standard [14] that monad maps $\mathsf{T} \to \mathsf{T}_S$ correspond to extensions to the left in:

$$\begin{array}{ccc} Set & \xrightarrow{F^{\mathsf{T}_S}} & Kl(\mathsf{T}_S) \\ {\scriptstyle F^{\mathsf{T}}}\downarrow & \nearrow & \\ Kl(\mathsf{T}) & & \end{array} \qquad \begin{array}{ccc} Set & \xrightarrow{(-)\cdot S} & \mathcal{C}_S \\ {\scriptstyle F_{\mathbb{T}}}\downarrow & \nearrow & \\ Kl(\mathbb{T}) & & \end{array} .$$

Now the Kleisli category $Kl(\mathsf{T}_S)$ of the linear-use state monad is isomorphic to the category $\mathcal{C}_S$ whose objects are sets, and whose maps $A \to B$ are $\mathcal{C}$-maps $A \cdot S \to B \cdot S$, via an isomorphism which identifies F^{T_S} with $(-) \cdot S \colon Set \to \mathcal{C}_S$. Similarly we have $Kl(\mathsf{T}) \cong Kl(\mathbb{T})$ under Set. So monad morphisms $\mathsf{T} \to \mathsf{T}_S$ correspond to extensions as right above: and these, by direct inspection, correspond to extensions as in (9). ◀

If here $\mathcal{C}$ is itself the Kleisli category $Kl(\mathbb{R})$ of a theory $\mathbb{R}$, then the linear-use state monad of $S \in Kl(\mathbb{R})$ is the monad $\mathsf{R}(- \times S)^S$ found as the commuting combination of the state monad for S with the monad R induced by $\mathbb{R}$ (cf. [8, Theorem 10]). Monad maps $\mathsf{T} \to \mathsf{R}(S \times -)^S$ were in [10] termed $\mathbb{R}$-*residual* $\mathbb{T}$-*runners*, and for these the preceding result specialises to:

▶ **Proposition 25.** *Let $\mathbb{T}$ and $\mathbb{R}$ be algebraic theories and let S be a set. The following are in bijective correspondence:*

1. *$\mathbb{R}$-residual $\mathbb{T}$-comodels $\boldsymbol{S}$ with underlying set S;*
2. *$\mathbb{R}$-residual $\mathbb{T}$-runners $[\![-]\!]^{\boldsymbol{S}} : \mathsf{T} \to \mathsf{R}(- \times S)^S$;*
3. *Functorial extensions of $(-) \times S \colon Set \to Set$ through the Kleisli categories of $\mathbb{T}$ and $\mathbb{R}$:*

$$\begin{array}{ccc} Set & \xrightarrow{(-)\times S} & Set \\ {\scriptstyle F_{\mathbb{T}}}\downarrow & & \downarrow{\scriptstyle F_{\mathbb{R}}} \\ Kl(\mathbb{T}) & \xrightarrow{(-)\cdot \boldsymbol{S}} & Kl(\mathbb{R}) \end{array} . \tag{10}$$

By putting together Propositions 23 and 25, we have an intuitive definition of *tensor product* for a residual comodel and a comodel, or for two residual comodels.

▶ **Definition 26** (Tensor product of residual comodels)**.** *Let $\mathbb{V}$, $\mathbb{T}$, $\mathbb{R}$ be theories; $\boldsymbol{M}$ an $\mathbb{R}$-comodel in $\mathcal{C}$; $\boldsymbol{S}$ an $\mathbb{R}$-residual $\mathbb{T}$-comodel; and $\boldsymbol{U}$ a $\mathbb{T}$-residual $\mathbb{V}$-comodel. The tensor product $\boldsymbol{S} \cdot \boldsymbol{M}$ is the $\mathbb{T}$-comodel in $\mathcal{C}$ classified by the composite of extensions to the left below, while the tensor product $\boldsymbol{U} \cdot \boldsymbol{S}$ is the $\mathbb{R}$-residual $\mathbb{V}$-comodel classified by the composite to the right:*

$$\begin{array}{ccccc} Set & \xrightarrow{(-)\times S} & Set & \xrightarrow{(-)\cdot M} & \mathcal{C} \\ {\scriptstyle F_{\mathbb{T}}}\downarrow & & {\scriptstyle F_{\mathbb{R}}}\downarrow & \nearrow_{(-)\cdot \boldsymbol{M}} & \\ Kl(\mathbb{T}) & \xrightarrow{(-)\cdot \boldsymbol{S}} & Kl(\mathbb{R}) & & \end{array} \qquad \begin{array}{ccccc} Set & \xrightarrow{(-)\times U} & Set & \xrightarrow{(-)\times S} & Set \\ {\scriptstyle F_{\mathbb{V}}}\downarrow & & {\scriptstyle F_{\mathbb{T}}}\downarrow & & \downarrow{\scriptstyle F_{\mathbb{R}}} \\ Kl(\mathbb{V}) & \xrightarrow{(-)\cdot \boldsymbol{U}} & Kl(\mathbb{T}) & \xrightarrow{(-)\cdot \boldsymbol{S}} & Kl(\mathbb{R}) \end{array} . \tag{11}$$

In particular, when $\mathcal{C} = Set$ and $\mathcal{C} = Top$, the tensor product $\boldsymbol{S} \cdot \boldsymbol{M}$ just defined specialises to the tensor products of Definitions 16 and 18 above.

▶ Remark 27. Here is another perspective on Definition 26. To the left of (11), the functor $(-) \cdot \boldsymbol{M}$ preserves copowers, and so lifts to a functor $Comod(\mathbb{T}, Kl(\mathbb{R})) \to Comod(\mathbb{T}, \mathcal{C})$, whose value at $\boldsymbol{S}$ is the tensor product $\boldsymbol{S} \cdot \boldsymbol{M}$. We can obtain $\boldsymbol{U} \cdot \boldsymbol{S}$ to the right similarly.

▶ Remark 28. While space precludes a full treatment here, we remark that the tensor product to the right of (11) allows us to re-find the *lazy composition* of intensional stream processors, described in [7, §4], as the unique map of $\mathbb{T}_A$-residual $\mathbb{T}_C$-comodels $\boldsymbol{I_{BC}} \cdot \boldsymbol{I_{AB}} \to \boldsymbol{I_{AC}}$

4 Extensional stream processors as a final bimodel

In Section 3, we characterised the set of intensional stream processors as a final $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel. In this section, we give the main result of the paper, characterising the set of extensional stream processors $Top(A^{\mathbb{N}}, B^{\mathbb{N}})$ as a final *bimodel* [4, 22, 3] for $\mathbb{T}_A$ and $\mathbb{T}_B$.

▶ **Definition 29** (Bimodel). *Let $\mathbb{T}$ and $\mathbb{R}$ be theories. An $\mathbb{R}$-$\mathbb{T}$-bimodel $\boldsymbol{K}$ is an $\mathbb{R}$-model $(\boldsymbol{K}, [\![-]\!]_{\boldsymbol{K}})$ endowed with $\mathbb{T}$-comodel structure $[\![-]\!]^{\boldsymbol{K}}$ in the category $Mod(\mathbb{R})$ of $\mathbb{R}$-models.*

The main difficulty in working with $\mathbb{R}$-$\mathbb{T}$-bimodels is handling copowers in $Mod(\mathbb{R})$. A simple case is that of *free* $\mathbb{R}$-models: a copower of free models is free, and so we have canonical isomorphisms $B \cdot \boldsymbol{R}(V) \cong \boldsymbol{R}(B \times V)$, which for convenience, we will assume are in fact *identities*, i.e., that the chosen copower $B \cdot \boldsymbol{R}(V)$ is $\boldsymbol{R}(B \times V)$. The $\mathbb{R}$-$\mathbb{T}$-bimodels with free underlying $\mathbb{R}$-model are easy to identify: they correspond precisely to $\mathbb{R}$-residual $\mathbb{T}$-comodels, where the $\mathbb{R}$-$\mathbb{T}$-bimodel $\boldsymbol{R}(\boldsymbol{S})$ corresponding to the $\mathbb{R}$-residual $\mathbb{T}$-comodel $\boldsymbol{S}$ has underlying $\mathbb{R}$-model $\boldsymbol{R}(S)$ and co-operations $[\![\sigma]\!]^{\boldsymbol{R}(\boldsymbol{S})} = \big([\![\sigma]\!]^{\boldsymbol{S}}\big)^{\dagger} \colon \boldsymbol{R}(S) \to \boldsymbol{R}(|\sigma| \times S)$, where $(-)^{\dagger}$ is the Kleisli extension operation of Lemma 6.

To understand what we gain by looking at bimodels with non-free underlying model, it is helpful to think in terms of quotients by bisimulations. If $\boldsymbol{S}$ is an $\mathbb{R}$-residual $\mathbb{T}$-comodel, then we could define (cf. [17, Definition 5.2]) a *bisimulation* on $\boldsymbol{S}$ to be an equivalence relation $\sim$ on S such that each co-operation $[\![\sigma]\!]^{\boldsymbol{S}}$ sends $\sim$-related states to $\approx$-related computations in $R(|\sigma| \times S)$, where $\approx$ is the congruence generated by $(i, s) \approx (i, s')$ whenever $s \sim s'$. The definition ensures that the residual comodel structure descends to the quotient set $S/\!\sim$; however, this only gives the possibility of identifying operationally equivalent *states*, and not operationally equivalent *computations* over states. The following more generous definition rectifies this.

▶ **Definition 30** ($\mathbb{R}$-bisimulation). *Let $\mathbb{R}$ and $\mathbb{T}$ be theories. For any $\mathbb{R}$-congruence $\sim$ on the free model $\boldsymbol{R}(V)$ and any set B, the congruence $\sim_B$ on $\boldsymbol{R}(B \times V)$ is that generated by*

$$t \sim u \text{ in } R(V) \qquad \Longrightarrow \qquad t(\lambda s.(b, s)) \sim_B u(\lambda s.(b, s)) \text{ for all } b \in B.$$

If $\boldsymbol{S}$ is an $\mathbb{R}$-residual $\mathbb{T}$-comodel, then a congruence on $\boldsymbol{R}(S)$ is an $\mathbb{R}$-bisimulation *if the co-operations $[\![\sigma]\!]^{\boldsymbol{R}(\boldsymbol{S})} = \big([\![\sigma]\!]^{\boldsymbol{S}}\big)^{\dagger} \colon \boldsymbol{R}(S) \to \boldsymbol{R}(|\sigma| \times S)$ of the associated bimodel send $\sim$-congruent terms to $\sim_{|\sigma|}$-congruent terms.*

▶ **Lemma 31.** *Let $\boldsymbol{S}$ be an $\mathbb{R}$-residual $\mathbb{T}$-comodel and $\sim$ an $\mathbb{R}$-bisimulation on $\boldsymbol{R}(S)$. There is a unique structure of $\mathbb{R}$-$\mathbb{T}$-bimodel on the quotient $\mathbb{R}$-model $\boldsymbol{K} = \boldsymbol{R}(S)/\!\sim$ for which the the quotient map $q \colon \boldsymbol{R}(S) \twoheadrightarrow \boldsymbol{K}$ becomes a map of bimodels $\boldsymbol{R}(\boldsymbol{S}) \twoheadrightarrow \boldsymbol{K}$.*

Proof. If $\boldsymbol{R}(S)/\!\sim = \boldsymbol{K}$ then $\boldsymbol{R}(B \times S)/\!\sim_B$ is a presentation of the copower $B \cdot \boldsymbol{K}$. So the assumption that $\sim$ is an $\mathbb{R}$-bisimulation ensures that each basic co-operation of $\boldsymbol{R}(\boldsymbol{S})$ descends to a co-operation on $\boldsymbol{K}$, as to the left in:

$$\begin{array}{ccc} \boldsymbol{R}(S) & \xrightarrow{[\![\sigma]\!]^{\boldsymbol{R}(\boldsymbol{S})}} & \boldsymbol{R}(|\sigma| \times S) \\ {\scriptstyle q}\big\downarrow & & \big\downarrow{\scriptstyle q_{|\sigma|}} \\ \boldsymbol{K} & \xrightarrow{[\![\sigma]\!]^{\boldsymbol{K}}} & |\sigma| \cdot \boldsymbol{K} \end{array} \qquad\qquad \begin{array}{ccc} \boldsymbol{R}(S) & \xrightarrow{[\![t]\!]^{\boldsymbol{R}(\boldsymbol{S})}} & \boldsymbol{R}(V \times S) \\ {\scriptstyle q}\big\downarrow & & \big\downarrow{\scriptstyle q_V} \\ \boldsymbol{K} & \xrightarrow{[\![t]\!]^{\boldsymbol{K}}} & V \cdot \boldsymbol{K} \end{array}.$$

Since $\boldsymbol{R}(|\sigma| \times S)$ is the copower $|\sigma| \cdot \boldsymbol{R}(S)$, and the quotient map $q_{|\sigma|}$ is the copower $|\sigma| \cdot q$, it follows that the *derived* co-operations of $\boldsymbol{R}(\boldsymbol{S})$ descend to the corresponding *derived* co-operations on $\boldsymbol{K}$, as to the right above; whence the satisfaction of the $\mathbb{T}$-comodel equations for $\boldsymbol{R}(\boldsymbol{S})$ implies the corresponding satisfaction for $\boldsymbol{K}$. So $\boldsymbol{K}$ is an $\mathbb{R}$-$\mathbb{T}$-bimodel, and clearly this is the *unique* bimodel structure making q into a bimodel homomorphism. ◀

We can use this construction to explain how the passage from the final $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel to the final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel will collapse the intensionality we saw in Example 22.

▶ **Example 32.** Consider the two $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodels $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ of Example 22. While there is clearly no scope for quotienting by a bisimulation on the set of states $\{*\}$, we *can* non-trivially quotient each by a $\mathbb{T}_A$-bisimulation on $\boldsymbol{T}_A(*)$: namely the $\mathbb{T}_A$-congruence on $\boldsymbol{T}_A(*)$ generated by $* \sim \mathsf{read}(\lambda a.*)$. It is easy to see that this is a $\mathbb{T}_A$-bisimulation for both $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, and so we obtain quotient $\mathbb{T}_A$-$\mathbb{T}_B$-bimodels $\boldsymbol{T}_A(\boldsymbol{S}_1)/{\sim}$ and $\boldsymbol{T}_A(\boldsymbol{S}_2)/{\sim}$. In fact, these are visibly the *same* bimodel $\boldsymbol{K}$, with underlying $\mathbb{T}_A$-model the final model $\{*\}$, and with $\mathbb{T}_B$-comodel structure $[\![\mathsf{read}]\!]^{\boldsymbol{K}} : \boldsymbol{K} \to B \cdot \boldsymbol{K}$ given by the bth coproduct injection. So we have a cospan of bimodels $\boldsymbol{T}_A(\boldsymbol{S}_1) \twoheadrightarrow \boldsymbol{K} \twoheadleftarrow \boldsymbol{T}_A(\boldsymbol{S}_2)$, which in particular implies that the states $*$ of $\boldsymbol{T}_A(\boldsymbol{S}_1)$ and $\boldsymbol{T}_A(\boldsymbol{S}_2)$ must be identified in a *final* $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel.

This example provides supporting evidence for the main theorem we shall now prove: that the set $\mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$ of extensional stream processors is a final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel. To show this, we will first construct an adjunction as to the left in

$$\mathit{Mod}(\mathbb{T}_A) \underset{(-)\otimes \boldsymbol{A}^{\mathbb{N}}}{\overset{\mathit{Top}(\boldsymbol{A}^{\mathbb{N}},-)}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} \mathit{Top} \qquad \mathit{Comod}(\mathbb{T}_B, \mathit{Mod}(\mathbb{T}_A)) \underset{(-)\otimes \boldsymbol{A}^{\mathbb{N}}}{\overset{\mathit{Top}(\boldsymbol{A}^{\mathbb{N}},-)}{\underset{\longrightarrow}{\overset{\longleftarrow}{\top}}}} \mathit{Comod}(\mathbb{T}_B, \mathit{Top}) \quad (12)$$

We then show that *both* directions of this adjunction preserve coproducts, so in particular copowers; it will then follow that the adjunction to the left lifts to one as to the right on $\mathbb{T}_B$-comodels. The right adjoint of this lifted adjunction, like any right adjoint, will preserve terminal objects, and so must send the final topological $\mathbb{T}_B$-comodel $\boldsymbol{B}^{\mathbb{N}}$ to a final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel, with underlying set $\mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$.

To construct the adjunction to the left in (12) we apply a standard result of category-theoretic universal algebra (cf. [4, Theorem 2]). For self-containedness we give a full proof.

▶ **Proposition 33.** *Let $\mathcal{C}$ be a category with copowers and $\boldsymbol{S}$ a $\mathbb{T}$-comodel in $\mathcal{C}$. For any object $C \in \mathcal{C}$, the hom-set $\mathcal{C}(S, C)$ bears a structure of $\mathbb{T}$-model $\mathcal{C}(\boldsymbol{S}, C)$ with operations*

$$[\![\sigma]\!]_{\mathcal{C}(\boldsymbol{S},C)}\,(\lambda i.\, S \xrightarrow{f_i} C) = S \xrightarrow{[\![\sigma]\!]^{\boldsymbol{S}}} |\sigma| \cdot S \xrightarrow{\langle f_i \rangle_{i \in |\sigma|}} C \quad (13)$$

where $\langle f_i \rangle_{i \in |\sigma|}$ is the copairing of the f_i's. As C varies, this assignment underlies a functor $\mathcal{C}(\boldsymbol{S}, -) \colon \mathcal{C} \to \mathit{Mod}(\mathbb{T})$. If $\mathcal{C}$ is cocomplete, this functor has a left adjoint $(-)\otimes\boldsymbol{S} \colon \mathit{Mod}(\mathbb{T}) \to \mathcal{C}$.

Proof. For any $C \in \mathcal{C}$, the hom-functor $\mathcal{C}(-, C) \colon \mathcal{C}^{\mathrm{op}} \to \mathit{Set}$ sends copowers in $\mathcal{C}$ to powers in Set, and so sends $\mathbb{T}$-comodels in $\mathcal{C}$ to $\mathbb{T}$-models in Set. In particular, the $\mathbb{T}$-model induced by $\boldsymbol{S} \in \mathit{Comod}(\mathbb{T}, \mathcal{C})$ is $\mathcal{C}(\boldsymbol{S}, C)$ with the operations defined above. The functoriality of this assignment is clear, so it remains to exhibit the desired adjoint when $\mathcal{C}$ is cocomplete.

To this end, note that a $\mathbb{T}$-model homomorphism $\alpha \colon \boldsymbol{X} \to \mathcal{C}(\boldsymbol{S}, C)$ is equally a function $\alpha \colon X \to \mathcal{C}(S, C)$ such that, for all basic $\mathbb{T}$-operations σ and all $\vec{x} \in X^{|\sigma|}$, we have

$$S \xrightarrow{\alpha([\![\sigma]\!]_{\boldsymbol{X}}(\vec{x}))} C \qquad = \qquad S \xrightarrow{[\![\sigma]\!]^{\boldsymbol{S}}} |\sigma| \cdot S \xrightarrow{\langle \alpha(x_i) \rangle_{i \in |\sigma|}} C\,.$$

Transposing under $(-) \cdot S \dashv \mathcal{C}(S, -) \colon \mathcal{C} \to \mathit{Set}$, this is equally to give a map $\bar{\alpha} \colon X \cdot S \to C$ in $\mathcal{C}$ such that, for each basic $\mathbb{T}$-operation σ, postcomposition with $\bar{\alpha}$ equalises the two maps

$$X^{|\sigma|} \cdot S \xrightarrow{[\![\sigma]\!]_{\boldsymbol{X}} \cdot C} X \cdot S \qquad X^{|\sigma|} \cdot S \xrightarrow{X^{|\sigma|} \cdot [\![\sigma]\!]^{\boldsymbol{X}}} X^{|\sigma|} \cdot (|\sigma| \cdot S) \cong (X^{|\sigma|} \times |\sigma|) \cdot S \xrightarrow{\mathrm{ev} \cdot S} X \cdot S\,.$$

Thus, defining $\boldsymbol{X} \otimes \boldsymbol{S}$ to be the joint coequaliser of these parallel pairs as σ varies across the basic $\mathbb{T}$-operations, we have bijections $\mathcal{C}(\boldsymbol{X} \otimes \boldsymbol{S}, C) \cong \mathit{Mod}(\mathbb{T})(\boldsymbol{X}, \mathcal{C}(\boldsymbol{S}, C))$ natural in $C \in \mathcal{C}$, so that $\boldsymbol{X} \otimes \boldsymbol{S}$ is the value at $\boldsymbol{X}$ of the desired left adjoint $(-) \otimes \boldsymbol{S}$. ◀

▶ Remark 34. Again, the final part of this result expresses an abstract fact: $Mod(\mathbb{T})$ is the *free cocomplete category containing a comodel of* $\mathbb{T}$.

In particular, we may apply the preceding result when $\mathcal{C}$ is the cocomplete category Top and $\boldsymbol{S}$ is the final topological $\mathbb{T}_A$-comodel $\boldsymbol{A}^{\mathbb{N}}$ to obtain an adjunction as to the left in (12). We now show that both directions of this adjunction preserve coproducts, and so in particular copowers. Since left adjoints always preserve colimits, there is only work to do for the right adjoint $Top(\boldsymbol{A}^{\mathbb{N}}, -)\colon Top \to Mod(\mathbb{T}_A)$. First we spell out that, on objects, this functor acts by taking a space C to the set of continuous functions $Top(A^{\mathbb{N}}, C)$, under the A-ary magma structure split that takes a family $(f_a : a \in A)$ of functions to the function $\mathsf{split}(\vec{f})$ with

$$\mathsf{split}(\vec{f})(\vec{a}) = f_{a_0}(\partial\vec{a}) \ . \tag{14}$$

In other words, $\mathsf{split}(\vec{f})$ consumes the first token a_0 of its input and then continues as f_{a_0} on the rest of its input; note that split is in fact invertible, with inverse given by the function $\mathsf{split}^{-1}(f) = (f(a{-}) : a \in A)$. This describes the action of $Top(\boldsymbol{A}^{\mathbb{N}}, -)\colon Top \to Mod(\mathbb{T}_A)$ on objects; on morphisms, it simply acts by postcomposition.

The following result is the main piece of serious work needed to complete our result; it refines the topological arguments described in [7, Theorem 2.1], and used there to construct the reification function for intensional stream processors.

▶ **Proposition 35.** *The functor* $Top(\boldsymbol{A}^{\mathbb{N}}, -)\colon Top \to Mod(\mathbb{T}_A)$ *preserves coproducts.*

Proof. Given spaces $(X_i : i \in I)$, we have the coproduct injections $\iota_i\colon X_i \to \Sigma_i X_i$ in Top, and must show that the family of postcomposition maps

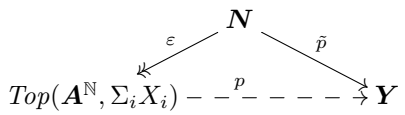
$$\big(\, \iota_i \circ (-)\colon Top(\boldsymbol{A}^{\mathbb{N}}, X_i) \to Top(\boldsymbol{A}^{\mathbb{N}}, \Sigma_i X_i) \,\big)_{i \in I} \tag{15}$$

constitute a coproduct cocone in $Mod(\mathbb{T}_A)$. We first show:

▶ **Lemma 36.** *The maps* (15) *are jointly epimorphic in* $Mod(\mathbb{T}_A)$.

Proof. We show that the sub-A-ary magma $\boldsymbol{M} \subseteq Top(\boldsymbol{A}^{\mathbb{N}}, \Sigma_i X_i)$ generated by the image of the maps (15) is all of $Top(\boldsymbol{A}^{\mathbb{N}}, \Sigma_i X_i)$. So suppose not; then there exists some continuous $f\colon A^{\mathbb{N}} \to \Sigma_i X_i$ with $f \notin M$. Since we have $f = \mathsf{split}(\mathsf{split}^{-1}(f)) = \mathsf{split}(\lambda a.\, f(a{-}))$, we can find $a_0 \in A$ with $f(a_0{-}) \notin M$. Now repeating the argument with $f(a_0{-})$, we can find $a_1 \in A$ with $f(a_0 a_1{-}) \notin M$; and continuing in this fashion, making countably many dependent choices, we find some $\vec{a} \in A^{\mathbb{N}}$ such that for all n, the continuous function $f(a_0 a_1 \ldots a_n{-})\colon A^{\mathbb{N}} \to \Sigma_i X_i$ is not in M. In particular, none of these functions factor through any X_i; but as $f(\vec{a}) \in X_i$ for some i, this means there is *no* open neighbourhood of $\vec{a}$ which is mapped by f into the open neigbourhood X_i of $f(\vec{a})$, contradicting the continuity of f. ◀

Thus, to complete the proof, we need only show that, for a cocone $(p_i\colon Top(\boldsymbol{A}^{\mathbb{N}}, X_i) \to \boldsymbol{Y})_{i \in I}$ in $Mod(\mathbb{T}_A)$, there exists *some* map $p\colon Top(\boldsymbol{A}^{\mathbb{N}}, \Sigma_i X_i) \to \boldsymbol{Y}$ with $p \circ Top(\boldsymbol{A}^{\mathbb{N}}, \iota_i) = p_i$ for each i. To this end, consider the diagram of A-ary magmas



where $\boldsymbol{N} = (N, \nu)$ is the free A-ary magma generated by symbols $[f, i]$ for $i \in I$ and $f \in Top(\boldsymbol{A}^{\mathbb{N}}, X_i)$, where ε sends $[f, i]$ to $\iota_i f$ and where $\tilde{p}$ sends $[f, i]$ to $p_i(f)$. It suffices to exhibit a factorisation p of $\tilde{p}$ through ε as displayed. Now by the lemma above, ε is

epimorphic, and so the coequaliser of its kernel-congruence; so to obtain such a factorisation, it suffices to show that if $x, y \in N$ satisfy $\varepsilon(x) = \varepsilon(y)$, then they satisfy $\tilde{p}(x) = \tilde{p}(y)$. We do so by induction on the total number of magma operations ν in x and y:

- If $x = [f, i]$ and $y = [g, j]$ then $\varepsilon(x) = \varepsilon(y)$ says that $\iota_i f = \iota_j g$, which is possible only if $i = j$ and $f = g$. So $x = y$ and so certainly $\tilde{p}(x) = \tilde{p}(y)$.
- If $x = [f, i]$ and $y = \nu(\lambda a.\, y_a)$ then on taking $x_a = [f(a\text{–}), i]$ for each a, we get from $\varepsilon(x) = \varepsilon(y)$ that

$$\mathsf{split}(\lambda a.\, \varepsilon(x_a)) = \mathsf{split}(\lambda a.\, \iota_i f(a\text{–})) = \iota_i f = \varepsilon(x) = \varepsilon(y) = \varepsilon(\nu(\lambda a.\, y_a)) = \mathsf{split}(\lambda a.\, \varepsilon(y_a))$$

  which, since split is invertible, implies that $\varepsilon(x_a) = \varepsilon(y_a)$ for each $a \in A$. By induction, we have $\tilde{p}(x_a) = \tilde{p}(y_a)$ for each a, and so we have the desired equality:

$$\tilde{p}(x) = p_i(f) = \mathsf{split}(\lambda a.\, p_i(f(a\text{–}))) = \mathsf{split}(\lambda a.\, \tilde{p}(x_a)) = \mathsf{split}(\lambda a.\, \tilde{p}(y_a)) = \tilde{p}(\nu(\lambda a.\, y_a) = \tilde{p}(y)\ .$$

- The case where $x = \nu(\lambda a.\, x_a)$ and $y = [g, j]$ is dual.
- Finally, if $x = \nu(\lambda a.\, x_a)$ and $y = \nu(\lambda a.\, y_a)$, then from $\varepsilon(x) = \varepsilon(y)$ we get

$$\mathsf{split}(\lambda a.\, \varepsilon(x_a)) = \varepsilon(\nu(\lambda a.\, x_a)) = \varepsilon(x) = \varepsilon(y) = \varepsilon(\nu(\lambda a.\, y_a)) = \mathsf{split}(\lambda a.\, \varepsilon(y_a))$$

  and so by invertibility of split that $\varepsilon(x_a) = \varepsilon(y_a)$ for all a. By induction, $\tilde{p}(x_a) = \tilde{p}(y_a)$ for all a, and so the desired equality

$$\tilde{p}(x) = \tilde{p}(\nu(\lambda a.\, x_a) = \mathsf{split}(\lambda a.\, \tilde{p}(x_a)) = \mathsf{split}(\lambda a.\, \tilde{p}(y_a)) = \tilde{p}(\nu(\lambda a.\, y_a) = \tilde{p}(y)\ .$$ ◀

Using this result, we can conclude the argument as explained above. Since both adjoints to the left of (12) preserve coproducts, the adjunction lifts to an adjunction between categories of $\mathbb{T}_B$-comodels as to the right. In particular, the lifted right adjoint sends the final topological $\mathbb{T}_B$-comodel to a final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel, so giving our main theorem:

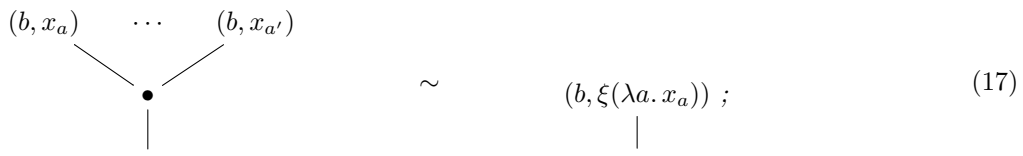
▶ **Theorem 37.** *For any sets A and B, the final $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel $\boldsymbol{E_{AB}}$ is given by the set of continuous functions $Top(A^{\mathbb{N}}, B^{\mathbb{N}})$ with the $\mathbb{T}_A$-model structure of (14), and with the $\mathbb{T}_B$-comodel structure map*

$$Top(\boldsymbol{A}^{\mathbb{N}}, B^{\mathbb{N}}) \xrightarrow{(g,n)\circ(-)} Top(\boldsymbol{A}^{\mathbb{N}}, B \cdot B^{\mathbb{N}}) \xrightarrow{\cong} B \cdot Top(\boldsymbol{A}^{\mathbb{N}}, B^{\mathbb{N}})\ , \tag{16}$$

whose first part is postcomposition with (1) and whose second part is the canonical isomorphism coming from the fact that $Top(\boldsymbol{A}^{\mathbb{N}}, \text{–}) \colon Top \to Mod(\mathbb{T}_A)$ preserves coproducts.

We now describe (16) more concretely, but first we describe copowers in $Mod(\mathbb{T}_A)$.

▶ **Lemma 38.** *For any $\mathbb{T}_A$-model $\boldsymbol{X} = (X, \xi)$ and set B, the copower $B \cdot \boldsymbol{X}$ may be found as either: (i) the quotient of $\boldsymbol{T}_A(B \times X)$ by the congruence which identifies*



(17)

or: (ii) the subset of $T_A(B \times X)$ on those A-ary branching trees where no non-trivial subtree has all its leaves labelled by the same element of B, with the $\mathbb{T}_A$-model structure map υ being that of $T_A(B \times X)$ except that $\upsilon(\lambda a.\, (b, x_a)) = (b, \xi(\lambda a.\, x_a))$.

Proof. (i) is the presentation $\boldsymbol{T}_A(B \times X)/{\sim_B}$ from Lemma 31 when $\sim$ is the congruence associated to the quotient $\mathrm{id}^\dagger \colon \boldsymbol{T}_A(X) \twoheadrightarrow \boldsymbol{X}$. As for (ii), these elements are the normal forms for the strongly normalising rewrite system obtained by applying (17) from left to right. ◀

Via presentation (i), we may thus describe (16) by associating to each $f \in \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$ a suitable tree in $T_A(B \times \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}}))$. For this, we use the identification of $B^{\mathbb{N}}$ with $B \cdot B^{\mathbb{N}}$ via $\vec{b} \mapsto (b_0, \partial\vec{b})$, together with Lemma 36, to see that f lies in the closure under the A-ary magma operation split on $\mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$ of the set of those $g \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$ for which $g(\vec{a})_0$ is constant. (This expresses algebraically the fact that, for each $\vec{a} \in A^{\mathbb{N}}$, there is some finite initial segment $a_0 \dots a_k$ of $\vec{a}$ such that $f(\vec{a}')_0 = f(\vec{a})_0$ whenever $a_0 \dots a_k = a'_0 \dots a'_k$.)

Choosing any such presentation of f gives a well-founded A-ary tree (encoding the applications of split) with leaves labelled by functions $g \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$ with $g(\vec{a})_0$ constant. Each such g is equally specified by the constant $b = g(\vec{a})_0$, and the function $h = \partial \circ g \colon A^{\mathbb{N}} \to B^{\mathbb{N}}$, so that our leaf labels are equally elements in $B \times \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}})$: so altogether we have an element of $T_A(B \times \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}}))$. Note that choosing a different presentation of f would yield a different element of $T_A(B \times \mathit{Top}(A^{\mathbb{N}}, B^{\mathbb{N}}))$; however, our theory ensures that these elements are congruent under (17), so yielding a well-defined element of $B \cdot \mathit{Top}(\boldsymbol{A}^{\mathbb{N}}, B^{\mathbb{N}})$.

5 Comparing intensional and extensional stream processors

To conclude the paper, we examine the unique maps from an arbitrary $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel to the final one, showing that these act as expected via the extent function of Definition 17; and, finally, we give a comodel-theoretic explanation of the reflection-reification pair $I_{AB} \leftrightarrows E_{AB}$.

We begin with a small refinement of Proposition 33.

▶ **Proposition 39** (cf. [20, Theorem 4.4])**.** *Let $\mathcal{C}$ be a cocomplete category and $\boldsymbol{S}$ a $\mathbb{T}$-comodel in $\mathcal{C}$. The functor $(-) \otimes \boldsymbol{S}$ of Proposition 33 may be chosen to render commutative the following diagram, whose top edge is as in* (9)*, and whose left edge is as in Definition 7:*

$$\begin{array}{ccc} \mathit{Kl}(\mathbb{T}) & \xrightarrow{(-)\cdot\boldsymbol{S}} & \mathcal{C} \\ {\scriptstyle I_{\mathbb{T}}}\downarrow & \nearrow_{(-)\otimes\boldsymbol{S}} & \\ \mathit{Mod}(\mathbb{T}) & & \end{array} \quad . \tag{18}$$

Proof. For a free $\mathbb{T}$-model $\boldsymbol{T}(V)$, we have natural bijections $\mathit{Mod}(\mathbb{T})(\boldsymbol{T}(V), \mathcal{C}(\boldsymbol{S}, C)) \cong \mathit{Set}(V, \mathcal{C}(S, C)) \cong \mathcal{C}(V \cdot S, C)$, and so we may take $\boldsymbol{T}(V) \otimes \boldsymbol{S} = V \cdot S$. This makes (18) commute on objects. On morphisms, given $\theta^\dagger \colon \boldsymbol{T}(V) \to \boldsymbol{T}(W)$ in $\mathit{Mod}(\mathbb{T})$, its image $\theta^\dagger \otimes \boldsymbol{S} \colon V \cdot S \to W \cdot S$ under $(-) \otimes \boldsymbol{S}$ is, by adjointness, the unique map making:

$$\begin{array}{ccccc} \mathcal{C}(W \cdot S, C) & \xrightarrow{\cong} & \mathit{Set}(W, \mathcal{C}(S, C)) & \xrightarrow{\cong} & \mathit{Mod}(\mathbb{T})(\boldsymbol{T}W, \mathcal{C}(\boldsymbol{S}, C)) \\ {\scriptstyle (-)\circ(\theta^\dagger\otimes\boldsymbol{S})}\downarrow & & \vdots & & \downarrow{\scriptstyle (-)\circ\theta^+} \\ \mathcal{C}(V \cdot S, C) & \xrightarrow{\cong} & \mathit{Set}(V, \mathcal{C}(S, C)) & \xrightarrow{\cong} & \mathit{Mod}(\mathbb{T})(\boldsymbol{T}V, \mathcal{C}(\boldsymbol{S}, C)) \end{array}$$

commute for all $C \in \mathcal{C}$. Now, the unique dotted map making the right square commute is, by the freeness of $\boldsymbol{T}(V)$, the function

$$(f_w \in \mathcal{C}(S, C) : w \in W) \qquad \mapsto \qquad (\llbracket\theta(v)\rrbracket_{\mathcal{C}(\boldsymbol{S},C)}\,(\vec{f}) : v \in V) \ .$$

But by induction on (13), we have $\llbracket\theta(v)\rrbracket_{\mathcal{C}(\boldsymbol{S},C)}\,(\vec{f}) = S \xrightarrow{\llbracket\theta(v)\rrbracket^{\boldsymbol{S}}} W \cdot S \xrightarrow{\langle f_i\rangle_{i\in|\sigma|}} C$; whence we must have $\theta^\dagger \otimes \boldsymbol{S} = \langle\llbracket\theta(v)\rrbracket^{\boldsymbol{S}}\rangle_{v\in V} = \theta \cdot \boldsymbol{S}$ as desired. ◀

We now characterise the unique maps to $\boldsymbol{E_{AB}}$ from bimodels induced by residual comodels.

▶ **Proposition 40.** *Let $\boldsymbol{S}$ be a $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel. The unique bimodel map from the associated bimodel $f\colon \boldsymbol{T}_A(\boldsymbol{S}) \to \boldsymbol{E_{AB}}$ is $(\lambda s.e(s,-))^\dagger$, the homomorphic extension of the currying $S \to Top(A^{\mathbb{N}}, B^{\mathbb{N}})$ of the extent function of Definition 17.*

Proof. Since $(-)\otimes \boldsymbol{A}^{\mathbb{N}}\colon Mod(\mathbb{T}_A) \to Top$ restricts back along $I_{\mathbb{T}_A}$ to $(-)\cdot \boldsymbol{A}^{\mathbb{N}}\colon Kl(\mathbb{T}_A) \to Top$, its lifting to a functor on $\mathbb{T}_B$-comodels must, by Remark 27, restrict along $I_{\mathbb{T}_A}$ to the tensor product of Definition 26. So the unique $\mathbb{T}_B$-comodel map $\boldsymbol{T}_A(\boldsymbol{S}) \otimes \boldsymbol{A}^{\mathbb{N}} \to \boldsymbol{B}^{\mathbb{N}}$ must be the extent map $\boldsymbol{S}\cdot\boldsymbol{A}^{\mathbb{N}} \to \boldsymbol{B}^{\mathbb{N}}$ of Definition 17. By the proof of Proposition 39, transposing this to a bimodel map $\boldsymbol{T}_A(\boldsymbol{S}) \to \boldsymbol{E_{AB}}$ is achieved by currying and extending homomorphically. ◀

We now do the same for the unique maps to $\boldsymbol{E_{AB}}$ from *arbitrary* $\mathbb{T}_A$-$\mathbb{T}_B$-bimodels. To do so, we show that every such bimodel arises in a canonical way from the construction of Lemma 31. Note that this is *not* true for arbitrary theories $\mathbb{R}$ and $\mathbb{T}$.

▶ **Lemma 41.** *Let $\boldsymbol{K}$ be a $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel. The composite*

$$\gamma = K \xrightarrow{[\![\mathsf{read}]\!]^{\boldsymbol{K}}} B\cdot K \xrightarrow{\subseteq} T_A(B\times K)$$

where we take $B\cdot\boldsymbol{K} \subseteq T_A(B\times K)$ as in Lemma 38(ii), endows K with the structure of a $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel $\check{\boldsymbol{K}}$. The congruence on $\boldsymbol{T}_A(K)$ generating the $\mathbb{T}_A$-model quotient map $\mathrm{id}^\dagger_K\colon \boldsymbol{T}_A(K) \twoheadrightarrow \boldsymbol{K}$ is a $\mathbb{T}_A$-bisimulation for $\check{\boldsymbol{K}}$ and the quotient bimodel is precisely $\boldsymbol{K}$.

Proof. Only the final sentence requires any verification; it will follow if we can show that the square of $\mathbb{T}_A$-model maps to the left below is commutative:

$$\begin{array}{ccc} \boldsymbol{T}_A(K) & \xrightarrow{\gamma^\dagger} & \boldsymbol{T}_A(B\times K) \\ {\scriptstyle \mathrm{id}^\dagger_K}\downarrow & & \downarrow{\scriptstyle B\cdot\mathrm{id}^\dagger_K} \\ \boldsymbol{K} & \xrightarrow{[\![\mathsf{read}]\!]^{\boldsymbol{K}}} & B\cdot\boldsymbol{K} \end{array} \qquad \begin{array}{ccccc} K & \xrightarrow{[\![\mathsf{read}]\!]^{\boldsymbol{K}}} & B\cdot K & \xrightarrow{\iota} & T_A(B\times K) \\ {\scriptstyle \mathrm{id}}\downarrow & & & & \downarrow{\scriptstyle B\cdot\mathrm{id}^\dagger_K} \\ K & & \xrightarrow{[\![\mathsf{read}]\!]^{\boldsymbol{K}}} & & B\cdot K \end{array}$$

which by freeness will happen just when the diagram to the right also commutes. But the map $B\cdot\mathrm{id}^\dagger_K$ therein is the quotient map by the congruence of (17), of which ι must be a section since it selects a family of equivalence-class representatives. ◀

If $\boldsymbol{K}$ is a bimodel, then $\check{\boldsymbol{K}}$ is the *maximally lazy* realisation of $\boldsymbol{K}$ as a residual comodel, wherein the program associated to each state $k \in \boldsymbol{K}$ reads the absolute minimum number of input A-tokens required to determine the next output B-token, with all subsequent reading from A handed off (via the $\mathbb{T}_A$-model structure on $\boldsymbol{K}$) to the continuation state.

▶ **Proposition 42.** *Let $\boldsymbol{K}$ be a $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel. The image of $k \in \boldsymbol{K}$ under the unique bimodel map $\boldsymbol{K} \to \boldsymbol{E_{AB}}$ is the continuous function $e(k,-)\colon A^{\mathbb{N}} \to B^{\mathbb{N}}$, where e is the topological extent function associated to the $\mathbb{T}_A$-residual $\mathbb{T}_B$-comodel $\check{\boldsymbol{K}}$ of Lemma 41.*

Proof. By Lemma 41 we have a quotient map of bimodels $\mathrm{id}^\dagger_K\colon \boldsymbol{T}_A(\check{\boldsymbol{K}}) \twoheadrightarrow \boldsymbol{K}$ which of necessity fits into a commuting triangle

$$\begin{array}{ccc} \boldsymbol{T}_A(\check{\boldsymbol{K}}) & \xrightarrow{\mathrm{id}^\dagger_K} & \boldsymbol{K} \\ & {\scriptstyle !}\searrow \quad \swarrow{\scriptstyle !} & \\ & \boldsymbol{E_{AB}} & \end{array}.$$

The left edge of this triangle is the map identified in the preceding proposition. Thus, tracing elements $k \in K \subseteq T_A(K)$ around the two sides of this triangle yields the result. ◀

Finally, we give use the above results to give a comodel-theoretic reconstruction of the reflection-reification pair. We already defined $\mathsf{reflect}\colon I_{AB} \to E_{AB}$ in Definition 21. In the other direction, we define the *reification function* $\mathsf{reify}\colon E_{AB} \to I_{AB}$ as the underlying map of the unique residual comodel map $\check{\boldsymbol{E}}_{\boldsymbol{AB}} \to \boldsymbol{I}_{\boldsymbol{AB}}$.

▶ **Proposition 43.** *We have* $\mathsf{reflect} \circ \mathsf{reify} = \mathrm{id}_{E_{AB}}$.

Proof. By Proposition 40, the unique $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel map $\boldsymbol{T}_A(\boldsymbol{I}_{\boldsymbol{AB}}) \to \boldsymbol{E}_{\boldsymbol{AB}}$ is $\mathsf{reflect}^\dagger$, while by Lemma 41, $\mathrm{id}^\dagger_{E_{AB}}$ is the unique bimodel $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel map $\boldsymbol{T}_A(\check{\boldsymbol{E}}_{\boldsymbol{AB}}) \to \boldsymbol{E}_{\boldsymbol{AB}}$. So we have a (necessarily commuting) triangle of $\mathbb{T}_A$-$\mathbb{T}_B$-bimodel maps:

$$\begin{array}{ccc} \boldsymbol{T}_A(\check{\boldsymbol{E}}_{\boldsymbol{AB}}) & \xrightarrow{\boldsymbol{T}_A(\mathsf{reify})} & \boldsymbol{T}_A(\boldsymbol{I}_{\boldsymbol{AB}}) \\ & {\scriptstyle \mathrm{id}^\dagger_{E_{AB}}} \searrow \quad \swarrow {\scriptstyle \mathsf{reflect}^\dagger} & \\ & \boldsymbol{E}_{\boldsymbol{AB}} & \end{array}$$

and precomposing with $\eta\colon E_{AB} \to T_A(E_{AB})$ yields the result. ◀

As the notation suggests, the composite $\mathsf{reify} \circ \mathsf{reflect}$ implements *normalisation-by-evaluation* for intensional stream processors, where the normal forms are the maximally lazy elements of I_{AB}. For instance, the trees $t_1, t_2 \in I_{AB}$ of Example 22 will both normalise to t_1.

References

1 Danel Ahman and Andrej Bauer. Runners in action. In *Programming Languages and Systems*, volume 12075 of *Lecture Notes in Computer Science*, pages 29–55. Springer, 2020.

2 Mathieu Anel and André Joyal. Sweedler theory for (co)algebras and the bar-cobar constructions. Preprint, available as `arXiv:1309.6952`, 2013.

3 George M. Bergman and Adam O. Hausknecht. *Co-groups and co-rings in categories of associative rings*, volume 45 of *Mathematical Surveys and Monographs*. American Mathematical Society, 1996.

4 Peter Freyd. Algebra valued functors in general and tensor products in particular. *Colloquium Mathematicum*, 14:89–106, 1966.

5 Richard Garner. The costructure-cosemantics adjunction for comodels for computational effects. Preprint, available as arXiv:2011.14520, 2020.

6 Sergey Goncharov, Stefan Milius, and Alexandra Silva. Toward a uniform theory of effectful state machines. *ACM Transactions on Computational Logic*, 21, 2020.

7 Peter Hancock, Dirk Pattinson, and Neil Ghani. Representations of stream processors using nested fixed points. *Logical Methods in Computer Science*, 5:3:9, 17, 2009.

8 Martin Hyland, Gordon Plotkin, and John Power. Combining effects: sum and tensor. *Theoretical Computer Science*, 357:70–99, 2006.

9 Bart Jacobs and Sam Staton. De Finetti's construction as a categorical limit. In *Coalgebraic methods in computer science*, volume 12094 of *Lecture Notes in Computer Science*, pages 90–111. Springer, 2020. `doi:10.1007/978-3-030-57201-3_6`.

10 Shin-ya Katsumata, Exequiel Rivas, and Tarmo Uustalu. Interaction laws of monads and comonads. Preprint, available as arXiv:1912.13477, 2019.

11 Clemens Kupke and Raul Andres Leal. Characterising behavioural equivalence: three sides of one coin. In *Algebra and coalgebra in computer science*, volume 5728 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2009.

12 Paul Blain Levy. *Call-by-push-value*, volume 2 of *Semantic Structures in Computation*. Kluwer, 2003.

13 Fred E. J. Linton. Some aspects of equational categories. In *Conference on Categorical Algebra (La Jolla, 1965)*, pages 84–94. Springer, 1966.

14 Jean-Pierre Meyer. Induced functors on categories of algebras. *Mathematische Zeitschrift*, 142:1–14, 1975.

15 Rasmus Ejlers Møgelberg and Sam Staton. Linear usage of state. *Logical Methods in Computer Science*, 10:1:17, 52, 2014.

16 Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

17 Dirk Pattinson and Lutz Schröder. Sound and complete equational reasoning over comodels. In *The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI)*, volume 319 of *Electronic Notes in Theoretical Computer Science*, pages 315–331. Elsevier, 2015.

18 Dirk Pattinson and Lutz Schröder. Program equivalence is coinductive. In *Proceedings of the 31st Annual ACM-IEEE Symposium on Logic in Computer Science (LICS 2016)*, page 10. ACM, 2016.

19 Gordon Plotkin and John Power. Notions of computation determine monads. In *Foundations of software science and computation structures (Grenoble, 2002)*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356. Springer, Berlin, 2002.

20 Gordon Plotkin and John Power. Tensors of comodels and models for operational semantics. *Electronic Notes in Theoretical Computer Science*, 218:295–311, 2008.

21 John Power and Olha Shkaravska. From comodels to coalgebras: state and arrays. In *Proceedings of the Workshop on Coalgebraic Methods in Computer Science*, volume 106 of *Electronic Notes in Theoretical Computer Science*, pages 297–314. Elsevier, 2004.

22 D. O. Tall and G. C. Wraith. Representable functors and operations on rings. *Proceedings of the London Mathematical Society*, 20:619–643, 1970.

23 Tarmo Uustalu. Stateful runners of effectful computations. In *The 31st Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXI)*, volume 319 of *Electron. Notes Theor. Comput. Sci.*, pages 403–421. Elsevier Sci. B. V., Amsterdam, 2015.

24 Tarmo Uustalu and Niels Voorneveld. Algebraic and coalgebraic perspectives on interaction laws. In *Programming Languages and Systems*, volume 12470 of *Lecture Notes in Computer Science*, pages 186–205. Springer, 2020.

A Coinductive Version of Milner's Proof System for Regular Expressions Modulo Bisimilarity

Clemens Grabmayer
Gran Sasso Science Institute, L'Aquila, Italy

Abstract

By adapting Salomaa's complete proof system for equality of regular expressions under the language semantics, Milner (1984) formulated a sound proof system for bisimilarity of regular expressions under the process interpretation he introduced. He asked whether this system is complete. Proof-theoretic arguments attempting to show completeness of this equational system are complicated by the presence of a non-algebraic rule for solving fixed-point equations by using star iteration.

We characterize the derivational power that the fixed-point rule adds to the purely equational part Mil^- of Milner's system Mil: it corresponds to the power of coinductive proofs over Mil^- that have the form of finite process graphs with the loop existence and elimination property LEE. We define a variant system cMil by replacing the fixed-point rule in Mil with a rule that permits LEE-shaped circular derivations in Mil^- from previously derived equations as a premise. With this rule alone we also define the variant system CLC for combining LEE-shaped coinductive proofs over Mil^-. We show that both cMil and CLC have proof interpretations in Mil, and vice versa. As this correspondence links, in both directions, derivability in Mil with derivation trees of process graphs, it widens the space for graph-based approaches to finding a completeness proof of Milner's system.

2012 ACM Subject Classification Theory of computation → Process calculi

Keywords and phrases regular expressions, process theory, bisimilarity, coinduction, proof theory

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.16

Related Version The report version [9] contains an extended appendix, which includes the detailed justification of the proof transformation from the coinductive system cMil to Milner's system Mil. *Extended Version*: `https://arxiv.org/abs/2108.13104`

Funding The completion of this research was supported by the PRIN project IT MATTERS – *Methods and Tools for Trustworthy Smart Systems* (with project ID: 2017FTXR7S_005).

Acknowledgements I want to thank Luca Aceto for his comments on the introduction. I am also very grateful to the anonymous reviewers for their careful reading, for spotting distracting oversights, and for asking for additional clarifications, such as concerning the motivation of coinductive proofs.

1 Introduction

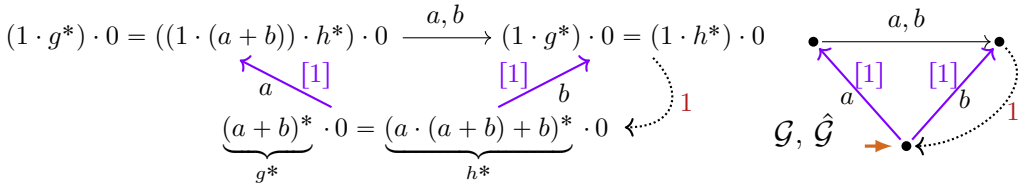
Milner [13] (1984) defined a process semantics for regular expressions as process graphs: the interpretation of 0 is deadlock, of 1 is successful termination, letters a are atomic actions, the operators $+$ and $\cdot$ stand for choice and concatenation of processes, and (unary) Kleene star $(\cdot)^*$ represents iteration with the option to terminate successfully before each execution of the iteration body. To disambiguate the use of regular expressions for denoting processes and comparing them via bisimilarity, Milner called them "star expressions". Using bisimilarity to identify processes with the same behavior, he was interested in an axiomatization of equality of "star behaviors", which are bisimilarity equivalence classes of star-expression processes. He adapted Salomaa's complete proof system [14] for language equivalence on regular expressions to a system Mil that is sound for equality of denoted star behaviors. He left completeness as a question, because he recognized that Salomaa's proof route cannot be followed directly.

Specifically, Milner gave an example showing that systems of guarded equations with star expressions cannot be solved by star expressions in general. Even if such a system is solvable, the absence from Mil of the *left*-distributivity law $x \cdot (y + z) = x \cdot y + x \cdot z$ in Salomaa's system

(it is not sound under bisimilarity) frequently prevents applications of the fixed-point rule RSP* in Mil like in an extraction procedure from Salomaa's proof. But if RSP* is replaced in Mil by a general unique-solvability rule scheme for guarded systems of equations (see Def. 2.4), then a complete system arises (noted in [6]). Therefore completeness of Mil hinges on whether the fixed-point rule RSP* enables to prove equal any two star-expression solutions of a given guarded system of equations, on the basis of the purely equational part Mil$^-$ of Mil.

As a stepping stone for tackling this difficult question, we here characterize the derivational power that the fixed-point rule RSP* adds to the subsystem Mil$^-$ of Mil. We do so by means of "coinductive proofs" whose shapes have the "loop existence and elimination property" LEE from [11]. This property stems from the interpretation of (1-free) star expressions, which is defined by induction on syntax trees, creating a hierarchy of "loop subgraphs". Crucially for our purpose, guarded systems of equations that correspond to finite process graphs with LEE *are* uniquely solvable modulo provability in Mil$^-$. The reason is that process graphs with LEE, which need not be in the image of the process interpretation, are amenable to applying *right*-distributivity and the rule RSP* for an extraction procedure like in Salomaa's proof (see Section 5). These graphs can be expressed modulo bisimilarity by some star expression, which can be used to show that any two solutions modulo Mil$^-$ of a specification of LEE-shape are Mil-provably equal. This is a crucial step in the completeness proof by Fokkink and myself in [11] for the tailored restriction BBP of Milner's system Mil to "1-free" star expressions.

Thus motivated, we define a "LLEE-witnessed coinductive proof" as a process graph $\mathcal{G}$ with "layered" LEE (LLEE) whose vertices are labeled by equations between star expressions. The left- and the right-hand sides of the equations have to form a solution vector of a specification corresponding to the process graph $\mathcal{G}$. However, that specification needs to be satisfied only up to provability in Mil$^-$ from sound assumptions. Such coinductive derivations are typically circular, like the one below of the semantically valid equation $(a+b)^* \cdot 0 = (a \cdot (a+b)+b)^* \cdot 0$:



The process graph $\mathcal{G}$, which is given together with a labeling $\hat{\mathcal{G}}$ that is a "LLEE-witness" of $\mathcal{G}$ (the colored transitions with marking labels $[n]$, for $n \in \mathbb{N}^+$, indicate LLEE-structure, see Section 3), underlies the coinductive proof on the left (see Ex. A.1 in the Appendix for a justification). $\mathcal{G}$ is a "1-chart" that is, a process graph with 1-transitions that represent empty step processes. We depict 1-transitions as dotted arrows. For 1-charts, "1-bisimulation" is the adequate concept of bisimulation. We showed in [10, 8] that the process (chart) interpretation $\mathcal{C}(e)$ of a star expression e is the image of a 1-chart $\underline{\mathcal{C}}(e)$ with LLEE under a functional 1-bisimulation. In this example, $\mathcal{G} = \underline{\mathcal{C}}(h^* \cdot 0)$ maps by a functional 1-bisimulation to interpretations of both expressions in the conclusion. The correctness conditions for such coinductive proofs are formed by the requirement that the left-, and respectively, the right-hand sides of formal equations form "Mil$^-$-provable solutions" of the underlying process graph: an expression at a vertex v can be reconstructed, provably in Mil$^-$, from the transitions to, and the expressions at, immediate successor vertices of v. Crucially we establish in Section 5, by a generalization of arguments in [11, 12] using RSP*, that every LLEE-witnessed coinductive proof over Mil$^-$ can be transformed into a derivation in Mil with the same conclusion.

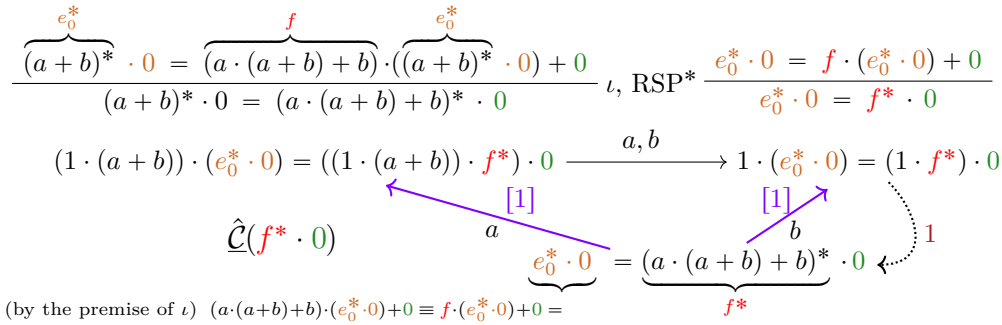


Figure 1 Mimicking an instance ι of the fixed-point rule RSP* (above) in Milner's system Mil = Mil$^-$+RSP* by a coinductive proof (below) over Mil$^-$+{premise of ι} with LLEE-witness $\underline{\hat{\mathcal{C}}}(f^*\cdot 0)$. We use different colors for indicating the expressions $e_0^*\cdot 0$, f, f^*, and 0 in the instance ι of RSP*.

This raises the question of whether the fixed-point rule RSP* of Mil adds any derivational power to Mil$^-$ that goes beyond those of LLEE-witnessed coinductive proofs over Mil$^-$, and if so, how far precisely. As our main result we show in Section 6 that every instance of the fixed-point rule RSP* can be mimicked by a LLEE-witnessed coinductive proof over Mil$^-$ in which also the premise of the rule may be used. It follows that the derivational power that RSP* adds to Mil$^-$ within Mil consists of iterating such LLEE-witnessed coinductive proofs along finite (meta-)prooftrees. The example in Fig. 1 (see Ex. A.2 in the Appendix for a justification) can give a first impression of the construction that we will use (in the proof of Lem. 6.2) to mimic instances of RSP*. Here this construction results in a coinductive proof that only differs slightly from the one with the same underlying LLEE-1-chart we saw earlier.

Based on these two proof transformations we obtain a theorem-equivalent, coinductive variant cMil of Mil by replacing RSP* with a rule that as one premise permits a LLEE-witnessed coinductive proof over Mil$^-$ plus the equations of other premises. We also define a theorem-equivalent system CLC ("combining LLEE-witnessed coinductive proofs") with this rule alone. While CLC only has LEE-shaped coinductive proofs over Mil$^-$ as formulas, we use a hybrid concept of formula in cMil that also permits equations between star expressions.

Additionally, we formulate proof systems $\overline{\text{cMil}}$ and CC that arise from cMil and CLC by dropping "LLEE-witnessed" as a requirement for coinductive proofs. These systems are (obviously) complete for bisimilarity of process interpretations, because they can mimic the unique solvability rule scheme for guarded systems of specifications mentioned before.

Our transformations are inspired by proof-theoretic interpretations in [4] between proof systems for recursive type equality by Amadio and Cardelli [1], and by Brandt and Henglein [3]. The transformation from cMil back to Mil is similar in kind to one we described in [5] from derivations in a coinductively motivated proof system for language equivalence between regular expressions to derivations in Salomaa's system [14] with a fixed-point rule similar to RSP*.

2 Process semantics for star expressions, and Milner's proof system

Here we fix terminology concerning star expressions, 1-charts, 1-bisimulations, we exhibit Milner's system (and a few variants), and recall the chart interpretation of star expressions.

Let A be a set of *actions*. The set $StExp(A)$ of *star expressions over actions in* A are strings that are defined by the following grammar:

$$e, e_1, e_2 ::= 0 \mid 1 \mid a \mid (e_1 + e_2) \mid (e_1 \cdot e_2) \mid (e^*) \qquad (\text{where } a \in A)$$

We will drop outermost brackets. We use e, f, g, h, possibly indexed and/or decorated, as syntactical variables for star expressions. We write $\equiv$ for syntactic equality between star

expressions denoted by such syntactical variables, and values of star expression functions, in a given context, but we permit $=$ in formal equations between star expressions. We denote by $Eq(A)$ the set of formal equations $e = f$ between two star expressions $e, f \in StExp(A)$.

We define sum expressions $\sum_{i=1}^{n} e_i$ inductively as 0 if $n = 0$, as e_1 if $n = 1$, and as $(\sum_{i=1}^{n-1} e_i) + e_n$ if $n > 0$, for $n \in \mathbb{N} = \{0, 1, 2, \ldots\}$. The *(syntactic) star height* $|e|_*$ of a star expression $e \in StExp(A)$ is the maximal nesting depth of stars in e, defined inductively by: $|0|_* := |1|_* := |a|_* := 0$, $|e_1 + e_2|_* := |e_1 \cdot e_2|_* := \max\{|e_1|_*, |e_2|_*\}$, and $|e^*|_* := 1 + |e|_*$.

A 1-*chart* is a 6-tuple $\langle V, A, 1, v_s, \to, \downarrow\rangle$ where V is a finite set of *vertices*, A is a set of *(proper) action labels*, $1 \notin A$ is the specified *empty step label*, $v_s \in V$ is the *start vertex* (hence $V \neq \varnothing$), $\to \subseteq V \times A(1) \times V$ is the *labeled transition relation*, where $A(1) := A \cup \{1\}$ is the set of action labels including 1, and $\downarrow \subseteq V$ is a set of *vertices with immediate termination*. In such a 1-chart, we call a transition in $\to \cap (V \times A \times V)$ (labeled by a *proper action* in A) a *proper transition*, and a transition in $\to \cap (V \times \{1\} \times V)$ (labeled by the *empty-step symbol* 1) a 1-*transition*. Reserving non-underlined action labels like $a, b, \ldots$ for proper actions, we use underlined action label symbols like $\underline{a}$ for actions labels in the set $A(1)$ that includes the label 1. We highlight in red transition labels that may involve 1.

We say that a 1-chart is *weakly guarded* if it does not contain cycles of 1-transitions. By a *chart* we mean a 1-chart that is 1-*free* in the sense that it does not contain 1-transitions.

Below we define the process semantics of regular (star) expressions as (1-free) charts, and hence as finite, rooted labeled transition systems, which will be compared with (1-)bisimilarity. The charts obtained correspond to non-deterministic finite-state automata that are obtained by iterating partial derivatives [2] of Antimirov (who did not aim at a process semantics).

▶ **Definition 2.1.** The *chart interpretation of* a star expression $e \in StExp(A)$ is the 1-transition free chart $\mathcal{C}(e) = \langle V(e), A, 1, e, \to \cap\, V(e), \downarrow \cap\, V(e)\rangle$, where $V(e)$ consists of all star expressions that are reachable from e via the labeled transition relation $\to \subseteq StExp(A) \times A \times StExp(A)$ that is defined, together with the immediate-termination relation $\downarrow \subseteq StExp(A)$, via derivability in the transition system specification (TSS) $\mathcal{T}(A)$, for $a \in A$, $e, e_1, e_2, e', e'_1, e'_2 \in StExp(A)$:

$$\frac{}{1\downarrow} \qquad \frac{e_i\downarrow}{(e_1 + e_2)\downarrow} \qquad \frac{e_1\downarrow \quad e_2\downarrow}{(e_1 \cdot e_2)\downarrow} \qquad \frac{}{(e^*)\downarrow}$$

$$\frac{}{a \xrightarrow{a} 1} \qquad \frac{e_i \xrightarrow{a} e'_i}{e_1 + e_2 \xrightarrow{a} e'_i} \qquad \frac{e_1 \xrightarrow{a} e'_1}{e_1 \cdot e_2 \xrightarrow{a} e'_1 \cdot e_2} \qquad \frac{e_1\downarrow \quad e_2 \xrightarrow{a} e'_2}{e_1 \cdot e_2 \xrightarrow{a} e'_2} \qquad \frac{e \xrightarrow{a} e'}{e^* \xrightarrow{a} e' \cdot e^*}$$

If $e \xrightarrow{a} e'$ is derivable in $\mathcal{T}(A)$, for $e, e' \in StExp(A)$, $a \in A$, then we say that e' is a *derivative* of e. If $e\downarrow$ is derivable in $\mathcal{T}(A)$, then we say that e *permits immediate termination*.

In Section 3 we define a refinement of this interpretation from [10] into a 1-chart interpretation. In both versions, (1-)charts obtained will be compared with respect to 1-bisimilarity that relates the behavior of "induced transitions" of 1-charts. By an *induced* a-*transition* $v \xrightarrow{(a]} w$, for a proper action $a \in A$, in a 1-chart $\underline{\mathcal{C}}$ we mean a path $v \xrightarrow{1} \cdots \xrightarrow{1} \cdot \xrightarrow{a} w$ in $\underline{\mathcal{C}}$ that consists of a finite number of 1-transitions that ends with a proper a-transition. By *induced termination* $v\downarrow^{(1)}$, for $v \in V$ we mean that there is a path $v \xrightarrow{1} \cdots \xrightarrow{1} \tilde{v}$ with $\tilde{v}\downarrow$ in $\underline{\mathcal{C}}$.

▶ **Definition 2.2** (1-bisimulation)**.** Let $\underline{\mathcal{C}}_i = \langle V_i, A, 1, v_{s,i}, \to_i, \downarrow_i\rangle$ be 1-charts, for $i \in \{1, 2\}$.

By a 1-*bisimulation between* $\underline{\mathcal{C}}_1$ *and* $\underline{\mathcal{C}}_2$ we mean a binary relation $B \subseteq V_1 \times V_2$ such that $\langle v_{s,1}, v_{s,2}\rangle \in B$, and for every $\langle v_1, v_2\rangle \in B$ the following three conditions hold:

(forth) $\quad \forall v'_1 \in V_1 \forall a \in A\big(\, v_1 \xrightarrow{(a]}_1 v'_1 \implies \exists v'_2 \in V_2\big(\, v_2 \xrightarrow{(a]}_2 v'_2 \wedge \langle v'_1, v'_2\rangle \in B\,\big)\big)$,

(back) $\quad \forall v'_2 \in V_2 \forall a \in A\big(\exists v'_1 \in V_1\big(\, v_1 \xrightarrow{(a]}_1 v'_1 \wedge \langle v'_1, v'_2\rangle \in B\,\big) \impliedby v_2 \xrightarrow{(a]}_2 v'_2\big)$,

(termination) $\quad v_1\downarrow_1^{(1)} \iff v_2\downarrow_2^{(1)}$.

We denote by $\underline{\mathcal{C}}_1 \leftrightarrows^{(1)} \underline{\mathcal{C}}_2$, and say that $\underline{\mathcal{C}}_1$ and $\underline{\mathcal{C}}_2$ are 1-*bisimilar*, if there is a 1-bisimulation between $\underline{\mathcal{C}}_1$ and $\underline{\mathcal{C}}_2$. We call 1-bisimilar (1-free) charts $\mathcal{C}_1$ and $\mathcal{C}_2$ *bisimilar*, and write $\mathcal{C}_1 \leftrightarrows \mathcal{C}_2$.

Let A be a set. The *basic proof system* $\mathcal{EL}(A)$ *of equational logic* for star expressions has as *formulas* the formal equations between star expressions in $Eq(A)$, and the following *rules*:

$$\frac{}{e=e}\ \text{Refl} \qquad \frac{e=f}{f=e}\ \text{Symm} \qquad \frac{e=f \qquad f=g}{e=g}\ \text{Trans} \qquad \frac{e=f}{C[e]=C[f]}\ \text{Cxt}$$

that is, the rules Refl (for reflexivity), and the rules Symm (for symmetry), Trans (for transitivity), and Cxt (for filling a context), where $C[]$ is a 1-hole star expression context.

By an $\mathcal{EL}$*-based system over* $StExp(A)$ (and *for star expressions over* A) we mean a proof system whose formulas are the formal equations in $Eq(A)$, and whose rules include the rules of the basic system $\mathcal{EL}(A)$ of equational logic (additionally, it may specify an arbitrary set of axioms). We will use $\mathcal{S}$ as syntactical variable for $\mathcal{EL}$-based proof systems.

Let $\mathcal{S}$ be an $\mathcal{EL}$-based proof system over $StExp(A)$, and $e_1, e_2 \in StExp(A)$. We permit to write $e_1 =_{\mathcal{S}} e_2$ for $\vdash_{\mathcal{S}} e_1 = e_2$, that is for the statement that there is a derivation without assumptions in $\mathcal{S}$ that has conclusion $e_1 = e_2$.

▶ **Definition 2.3** (sub-system, theorem equivalence, and theorem subsumption of proof systems)**.** Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be $\mathcal{EL}$-based proof systems over $StExp(A)$. We say that $\mathcal{S}_1$ is a *sub-system* of $\mathcal{S}_2$, denoted by $\mathcal{S}_1 \subseteq \mathcal{S}_2$, if every axiom of $\mathcal{S}_1$ is an axiom of $\mathcal{S}_2$, and every rule of $\mathcal{S}_1$ is also a rule of $\mathcal{S}_2$. We say that $\mathcal{S}_1$ *is theorem-subsumed by* $\mathcal{S}_2$, denoted by $\mathcal{S}_1 \precsim \mathcal{S}_2$, if whenever a formal equation $e_1 = e_2$ is derivable in $\mathcal{S}_1$ (without assumptions, by using only the rules and axioms of $\mathcal{S}_1$), then $e_1 = e_2$ is also derivable in $\mathcal{S}_2$. We say that $\mathcal{S}_1$ *and* $\mathcal{S}_2$ *are theorem-equivalent*, denoted by $\mathcal{S}_1 \sim \mathcal{S}_2$, if they have the same derivable equations.

▶ **Definition 2.4** (Milner's system Mil, variants and subsystems)**.** Let A be a set of actions.

By the proof system $\mathsf{Mil}^-(A)$ we mean the $\mathcal{EL}$-based proof system for star expressions over A with the following *axiom schemes*:

$$\begin{array}{rlrl}
(\text{assoc}(+)) & (e+f)+g = e+(f+g) & (\text{id}_{\text{l}}(\cdot)) & 1\cdot e = e \\
(\text{neutr}(+)) & e+0 = e & (\text{id}_{\text{r}}(\cdot)) & e\cdot 1 = e \\
(\text{comm}(+)) & e+f = f+e & (\text{deadlock}) & 0\cdot e = 0 \\
(\text{idempot}(+)) & e+e = e & (\text{rec}(^*)) & e^* = 1+e\cdot e^* \\
(\text{assoc}(\cdot)) & (e\cdot f)\cdot g = e\cdot(f\cdot g) & (\text{trm-body}(^*)) & e^* = (1+e)^* \\
(\text{r-distr}(+,\cdot)) & (e+f)\cdot g = e\cdot g + f\cdot g & &
\end{array}$$

where $e, f, g \in StExp(A)$, and with the *rules* of the system $\mathcal{EL}(A)$ of equational logic.

The *recursive specification principle for star iteration* RSP^*, the *unique solvability principle for star iteration* USP^*, and the *general unique solvability principle* USP are the rules:

$$\frac{e = f\cdot e + g}{e = f^*\cdot g}\ \text{RSP}^*\ (\text{if } f{\downarrow}) \qquad \frac{e_1 = f^*\cdot e_1 + g \qquad e_2 = f^*\cdot e_2 + g}{e_1 = e_2}\ \text{USP}^*\ (\text{if } f{\downarrow})$$

$$\frac{\left\{ e_{i,1} = \left(\sum_{j=1}^{n_i} f_{i,j}\cdot e_{j,1}\right) + g_i \right) \quad e_{i,2} = \left(\sum_{j=1}^{n_i} f_{i,j}\cdot e_{j,2}\right) + g_i \right) \Big\}_{i=1,\ldots,n}}{e_{1,1} = e_{1,2}}\ \text{USP}\ \begin{array}{l}(\text{if } f_{i,j}{\downarrow} \\ \text{for all } i,j)\end{array}$$

Milner's proof system $\mathsf{Mil}(A)$ is the extension of $\mathsf{Mil}^-(A)$ by adding the rule RSP^*. Its variant systems $\mathsf{Mil}'(A)$, and $\overline{\mathsf{Mil}}'(A)$, arise from $\mathsf{Mil}^-(A)$ by adding (instead of RSP^*) the rule USP^*, and respectively, the rule USP. $\mathsf{ACI}(A)$ is the system with the axioms for associativity, commutativity, and idempotency for $+$. We will keep the action set A implicit in the notation.

▶ **Proposition 2.5** (Milner, [13])**.** Mil *is sound for bisimilarity of chart interpretations. That is, for all* $e, f \in StExp(A)$ *it holds:* $\left(e =_{\mathsf{Mil}} f \implies \mathcal{C}(e) \leftrightarrows \mathcal{C}(f) \right)$.

▶ **Question 2.6** (Milner, [13]). Is Mil also complete for bisimilarity of process interpretations? That is, does for all $e, f \in StExp(A)$ the implication $(\, e =_{\mathsf{Mil}} f \iff \mathcal{C}(e) \leftrightarrows \mathcal{C}(f)\,)$ hold?

▶ **Definition 2.7** (provable solutions). Let $\mathcal{S}$ be an $\mathcal{EL}$-based proof system for star expressions over A that extends ACI. Let $\underline{\mathcal{C}} = \langle V, A, 1, v_s, \to, \downarrow \rangle$ be a 1-chart.

By a *star expression function on* $\underline{\mathcal{C}}$ we mean a function $s : V \to StExp(A)$ on the vertices of $\underline{\mathcal{C}}$. Let $v \in V$. We say that such a star expression function s on $\underline{\mathcal{C}}$ is an $\mathcal{S}$*-provable solution of* $\underline{\mathcal{C}}$ *at* v if it holds that $s(v) =_{\mathcal{S}} \tau_{\underline{\mathcal{C}}}(v) + \sum_{i=1}^{n} \underline{a_i} \cdot s(v_i)$, given the (possibly redundant) list representation $T_{\underline{\mathcal{C}}}(v) = \{ v \xrightarrow{\underline{a_i}} v_i \mid i \in \{1, \ldots, n\} \}$, of transitions from v in $\underline{\mathcal{C}}$ and where $\tau_{\underline{\mathcal{C}}}(v)$ is the *termination constant* $\tau_{\underline{\mathcal{C}}}(v)$ *of* $\underline{\mathcal{C}}$ *at* v defined as 0 if $v\not\downarrow$, and as 1 if $v\downarrow$. This definition does not depend on the specifically chosen list representation of $T_{\underline{\mathcal{C}}}(v)$, because $\mathcal{S}$ extends ACI, and therefore it contains the associativity, commutativity, and idempotency axioms for $+$.

By an $\mathcal{S}$*-provable solution of* $\underline{\mathcal{C}}$ (with *principal value* $s(v_s)$ at the start vertex v_s) we mean a star expression function s on $\underline{\mathcal{C}}$ that is an $\mathcal{S}$-provable solution of $\underline{\mathcal{C}}$ at every vertex of $\underline{\mathcal{C}}$.

3 Layered loop existence and elimination, and LLEE-witnesses

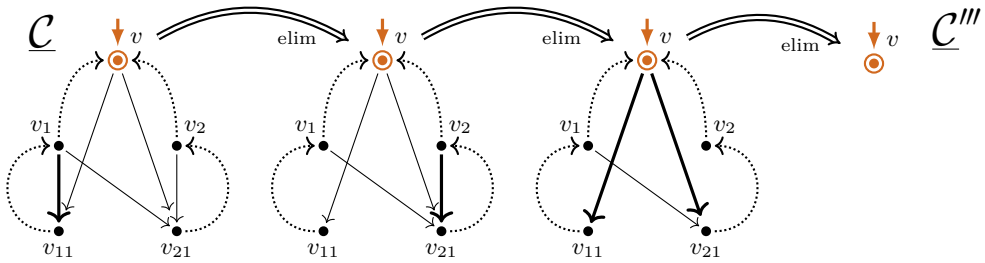
In this subsection we briefly recall principal definitions and statements from [11, 10]. We keep formalities to a minimum as necessary for our purpose (in particular for "LLEE-witnesses").

A 1-chart $\underline{\mathcal{L}} = \langle V, A, 1, v_s, \to, \downarrow \rangle$ is called a *loop 1-chart* if it satisfies three conditions:

(L1) There is an infinite path from the start vertex v_s.
(L2) Every infinite path from v_s returns to v_s after a positive number of transitions.
(L3) Immediate termination is only permitted at the start vertex, that is, $\downarrow \subseteq \{v_s\}$.

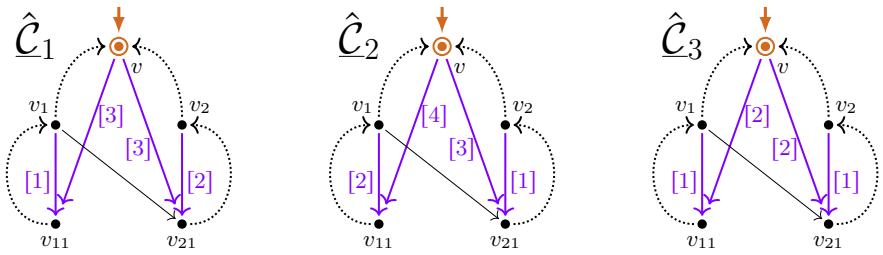
We call the transitions from v_s *loop-entry transitions*, and all other transitions *loop-body transitions*. A *loop sub-1-chart of* a 1-chart $\underline{\mathcal{C}}$ is a loop 1-chart $\underline{\mathcal{L}}$ that is a sub-1-chart of $\underline{\mathcal{C}}$ with some vertex $v \in V$ of $\underline{\mathcal{C}}$ as start vertex, such that $\underline{\mathcal{L}}$ is constructed, for a nonempty set U of transitions of $\underline{\mathcal{C}}$ from v, by all paths that start with a transition in U and continue onward until v is reached again (so the transitions in U are the loop-entry transitions of $\underline{\mathcal{L}}$).

The result of *eliminating a loop sub-1-chart* $\underline{\mathcal{L}}$ *from a 1-chart* $\underline{\mathcal{C}}$ arises by removing all loop-entry transitions of $\underline{\mathcal{L}}$ from $\underline{\mathcal{C}}$, and then also removing all vertices and transitions that become unreachable. We say that a 1-chart $\underline{\mathcal{C}}$ has the *loop existence and elimination property* (LEE) if the procedure, started on $\underline{\mathcal{C}}$, of repeated eliminations of loop sub-1-charts results in a 1-chart without an infinite path. If, in a successful elimination process from a 1-chart $\underline{\mathcal{C}}$, loop-entry transitions are never removed from the body of a previously eliminated loop sub-1-chart, then we say that $\underline{\mathcal{C}}$ satisfies *layered* LEE (LLEE), and is a LLEE-1-*chart*. While the property LLEE leads to a formally easier concept of "witness", it is equivalent to LEE. (For an example of a LEE-witness that is not layered, see further below on page 7.)



The picture above shows a successful run of the loop elimination procedure. In brown we highlight start vertices by ➔ , and immediate termination with a boldface ring by ⦿. The loop-entry transitions of loop sub-1-charts that are eliminated in the next step are marked in bold.

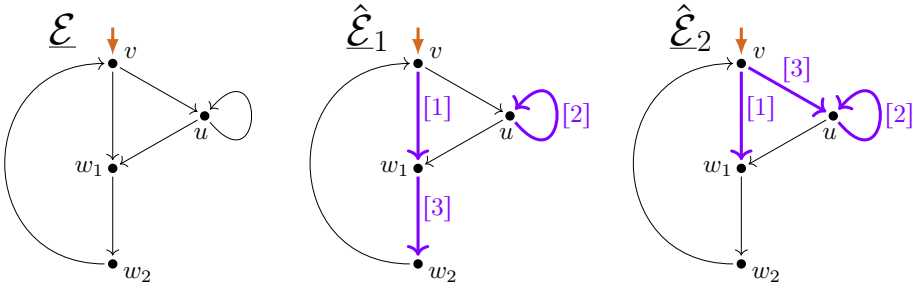
We have neglected action labels here, except for indicating 1-transitions by dotted arrows. Since the graph $\underline{\mathcal{C}}'''$ that is reached after three loop-subgraph elimination steps from the 1-chart $\underline{\mathcal{C}}$ does not have an infinite path, and no loop-entry transitions have been removed from a previously eliminated loop sub-1-chart, we conclude that $\underline{\mathcal{C}}$ satisfies LEE and LLEE.



A LLEE-*witness* $\hat{\underline{\mathcal{C}}}$ *of* a 1-chart $\underline{\mathcal{C}}$ is the recording of a successful run of the loop elimination procedure by attaching to a transition τ of $\underline{\mathcal{C}}$ the marking label n for $n \in \mathbb{N}^+$ (in pictures indicated as $[n]$, in steps as $\to_{[n]}$) forming a *loop-entry transition* if τ is eliminated in the n-th step, and by attaching marking label 0 to all other transitions of $\underline{\mathcal{C}}$ (in pictures neglected, in steps indicated as $\to_{\mathrm{bo}}$) forming a *body transition*. Formally, LLEE-witnesses arise as *entry/body-labelings* from 1-charts, and are charts in which the transition labels are pairs of action labels over A, and marking labels in $\mathbb{N}$. We say that a LLEE-witness $\hat{\underline{\mathcal{C}}}$ *is guarded* if all loop-entry transitions are proper, which means that they have a proper-action transition label.

The entry/body-labeling $\hat{\underline{\mathcal{C}}}_1$ above of the 1-chart $\underline{\mathcal{C}}$ is a LLEE-witness that arises from the run of the loop elimination procedure earlier above. The entry/body-labelings $\hat{\underline{\mathcal{C}}}_2$ and $\hat{\underline{\mathcal{C}}}_3$ of $\underline{\mathcal{C}}$ record two other successful runs of the loop elimination procedure of length 4 and 2, respectively, where for $\hat{\underline{\mathcal{C}}}_3$ we have permitted to eliminate two loop subcharts at different vertices together in the first step. The 1-chart $\underline{\mathcal{C}}$ above only has layered LEE-witnesses.

The situation is different for the 1-chart $\underline{\mathcal{E}}$ below:



The entry/body-labeling $\hat{\underline{\mathcal{E}}}_1$ of $\underline{\mathcal{E}}$ as above is a LEE-witness that is not layered: in the third loop sub-1-chart elimination step that is recorded in $\hat{\underline{\mathcal{E}}}_1$ the loop-entry transition from w_1 to w_2 is removed, which is in the body of the loop sub-1-chart at v with loop-entry transition from v to w_1 that (by that time) has already been removed in the first loop-elimination step as recorded in $\hat{\underline{\mathcal{E}}}_1$. By contrast, the entry/body-labeling $\hat{\underline{\mathcal{E}}}_2$ of $\underline{\mathcal{E}}$ above is a layered LEE-witness. In general it can be shown that every LEE-witness that is not layered can be transformed into a LLEE-witness of the same underlying 1-chart. Indeed, the step from $\hat{\underline{\mathcal{E}}}_1$ to $\hat{\underline{\mathcal{E}}}_2$ in the example above, which transfers the loop-entry transition marking label $[3]$ from the transition from w_1 to w_2 over to the transition from v to u, hints at the proof of this statement. However, we do not need this result, because we will be able to use the guaranteed existence of LLEE-witnesses (see Thm. 3.3) for the 1-chart interpretation below (see Def. 3.2).

In a LLEE-witness we denote by $v \rightsquigarrow w$, and by $w \leftsquigarrow v$, that w is in the body of the loop sub-1-chart at v, which means that there is a path $v \to_{[n]} v' \to_{\mathrm{bo}}^* w$ from v via a loop-entry transition and subsequent body transitions without encountering v again.

▶ **Lemma 3.1.** *The relations $\rightsquigarrow$ and $\to_{\mathrm{bo}}$ defined by a* LLEE*-witness $\hat{\mathcal{C}}$ of a* 1*-chart $\underline{\mathcal{C}}$ satisfy:*
(i) *$\leftsquigarrow^+$ is a well-founded, strict partial order on V.*
(ii) *$\leftarrow_{\mathrm{bo}}^+$ is a well-founded strict partial order on V.*

▶ **Definition 3.2** (1-chart interpretation of star expressions)**.** By the 1*-chart interpretation* $\underline{\mathcal{C}}(e)$ of a star expression e we mean the 1-chart that arises together with the entry/body-labeling $\hat{\underline{\mathcal{C}}}(e)$ as the e-rooted sub-LTS generated by $\{e\}$ according to the following TSS:

$$\frac{}{a \xrightarrow{a}_{\mathrm{bo}} 1} \qquad \frac{e_i \xrightarrow{a}_l E_i' \quad (i \in \{1,2\})}{e_1 + e_2 \xrightarrow{a}_{\mathrm{bo}} E_i'} \qquad \frac{e \xrightarrow{a}_l E' \quad (\text{if } nd^+(e))}{e^* \xrightarrow{a}_{[|e^*|_*]} E' * e^*} \qquad \frac{e \xrightarrow{a}_l E' \quad (\text{if } \neg nd^+(e))}{e^* \xrightarrow{a}_{\mathrm{bo}} E' * e^*}$$

$$\frac{E_1 \xrightarrow{a}_l E_1'}{E_1 \cdot e_2 \xrightarrow{a}_l E_1' \cdot e_2} \qquad \frac{E_1 \xrightarrow{a}_l E_1'}{E_1 * e_2^* \xrightarrow{a}_l E_1' * e_2^*} \qquad \frac{e_1\!\downarrow \qquad e_2 \xrightarrow{a}_l E_2'}{e_1 \cdot e_2 \xrightarrow{a}_{\mathrm{bo}} E_2'} \qquad \frac{e_1\!\downarrow}{e_1 * e_2^* \xrightarrow{1}_{\mathrm{bo}} e_2^*}$$

where $l \in \{\mathrm{bo}\} \cup \{[n] \mid n \in \mathbb{N}^+\}$, and star expressions using a "stacked" product operation $*$ are permitted, which helps to record iterations from which derivatives originate. Immediate termination for expressions of $\underline{\mathcal{C}}(e)$ is defined by the same rules as in Def. 2.1 (for star expressions only, preventing immediate termination for expressions with stacked product $*$). The condition $nd^+(e)$ means that e permits a positive length path to an expression f with $f\!\downarrow$. We use a projection function π that changes occurrences of stacked product $*$ into product $\cdot$.

▶ **Theorem 3.3** ([8, 10])**.** *For every $e \in StExp(A)$, (a) the entry/body-labeling $\hat{\underline{\mathcal{C}}}(e)$ of $\underline{\mathcal{C}}(e)$ is a* LLEE*-witness of $\underline{\mathcal{C}}(e)$, and (b) the projection function π defines a* 1*-bisimulation from the* 1*-chart interpretation $\underline{\mathcal{C}}(e)$ of e to the chart interpretation $\mathcal{C}(e)$ of e, and hence $\underline{\mathcal{C}}(e) \leftrightarrows^{(1)} \mathcal{C}(e)$.*

Lem. 3.5 below follows from the next lemma, whose proof we sketch in the appendix.

▶ **Lemma 3.4.** *$\pi(E) =_{\mathsf{Mil}^-} \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{i=1}^n \underline{a_i} \cdot \pi(E_i')$, given a list representation $T_{\underline{\mathcal{C}}(E)}(w) = \{E \xrightarrow{\underline{a_i}} E_i' \mid i \in \{1,\ldots,n\}\}$ of the transitions from E in $\underline{\mathcal{C}}(E)$.*

▶ **Lemma 3.5.** *For every star expression $e \in StExp(A)$ with* 1*-chart interpretation $\underline{\mathcal{C}}(e) = \langle V(e), A, 1, e, \to, \downarrow\rangle$ the star-expression function $s : V(e) \to StExp(A)$, $E \mapsto \pi(E)$ is a* Mil^-*-provable solution of $\underline{\mathcal{C}}(e)$ with principal value e.*

4 Coinductive version of Milner's proof system

In this section we motivate and define "coinductive proofs", introduce coinductive versions of Milner's system Mil, and establish first interconnections between these proof systems.

A finite 1-bisimulation between the 1-chart interpretations of star expressions e_1 and e_2 can be viewed as a proof of the statement that e_1 and e_2 define the same star behavior. This can be generalized by permitting finite 1*-bisimulations up to provability in* Mil, that is, finite relations B on star expressions for which pairs $\langle v_1, v_2\rangle \in B$ progress, via the (forth) and (back) conditions in Def. 2.2, to pairs $\langle v_1', v_2'\rangle$ in the (infinite) composed relation $=_{\mathsf{Mil}} \cdot B \cdot =_{\mathsf{Mil}}$. Now 1-bisimilarity up to $=_{\mathsf{Mil}}$ entails 1-bisimilarity of the 1-chart interpretations, and bisimilarity of chart interpretations, due to soundness of Mil (see Prop. 2.5). In order to link, later in Section 5, coinductive proofs with proofs in Milner's system Mil, we will be interested in 1-bisimulations up to $=_{\mathcal{S}}$ for systems $\mathcal{S}$ with $\mathsf{ACI} \subseteq \mathcal{S} \lesssim \mathsf{Mil}$ that have the form of LLEE-1-charts, which will guarantee such a connection. First we introduce a "LLEE-witnessed coinductive proof" as an equation-labeled, LLEE-1-chart $\underline{\mathcal{C}}$ that defines a 1-bisimulation up to $=_{\mathcal{S}}$ for $\mathcal{S}$ with $\mathsf{ACI} \subseteq \mathcal{S}$ from the left-/right-hand sides of equations on the vertices of $\underline{\mathcal{C}}$ (see Rem. 4.4).

▶ **Definition 4.1** ((LLEE-witnessed) coinductive proofs). Let $e_1, e_2 \in \mathit{StExp}(A)$ be star expressions, and let $\mathcal{S}$ be an $\mathcal{EL}$-based proof system for star expressions over A with $\mathsf{ACI} \subseteq \mathcal{S}$.

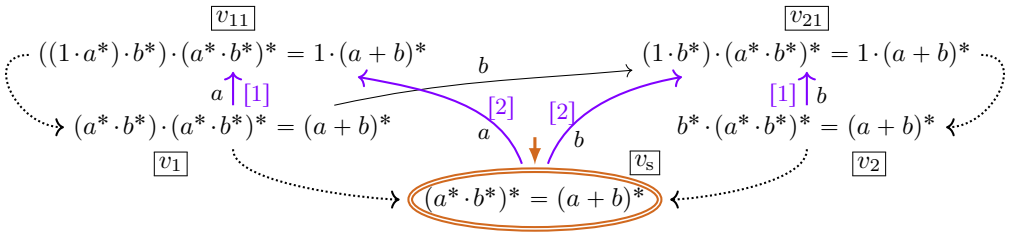
A *coinductive proof over* $\mathcal{S}$ *of* $e_1 = e_2$ is a pair $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ where $\underline{\mathcal{C}} = \langle V, A, 1, v_s, \to, \downarrow \rangle$ is a weakly guarded 1-chart, and $L : V \to \mathit{Eq}(A)$ a labeling function of vertices of $\underline{\mathcal{C}}$ by formal equations over A such that for the functions $L_1, L_2 : V \to \mathit{StExp}(A)$ that denote the star expressions $L_1(v)$, and $L_2(v)$, on the left-, and on the right-hand side of the equation $L(v)$, respectively, the following conditions hold:

(cp1) L_1 and L_2 are $\mathcal{S}$-provable solutions of $\underline{\mathcal{C}}$,

(cp2) $e_1 \equiv L_1(v_s)$ and $e_2 \equiv L_2(v_s)$.

By a LLEE-*witnessed coinductive proof* we mean a coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ where $\underline{\mathcal{C}}$ is a LLEE-1-chart. We denote by $e \overset{\text{coind}}{=\!=\!=}_{\mathcal{S}} f$ that there is a coinductive proof over $\mathcal{S}$ of $e = f$, and by $e \overset{\text{LLEE}}{=\!=\!=}_{\mathcal{S}} f$ that there is a LLEE-witnessed coinductive proof over $\mathcal{S}$ of $e = f$.

▶ **Example 4.2.** The statement $(a^* \cdot b^*)^* \overset{\text{LLEE}}{=\!=\!=}_{\mathsf{Mil}^-} (a+b)^*$ can be established by the following LLEE-witnessed coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ over Mil^- where $\underline{\mathcal{C}} = \underline{\mathcal{C}}((a^* \cdot b^*)^*)$ has the indicated LLEE-witness $\hat{\underline{\mathcal{C}}}((a^* \cdot b^*)^*)$ (see Thm. 3.3) where framed boxes contain vertex names:



Here we have drawn the 1-chart $\underline{\mathcal{C}}$ that carries the equations with its start vertex below in order to adhere to the prooftree intuition for the represented derivation, namely with the conclusion at the bottom. We will do so repeatedly also below. Solution correctness conditions for the left-hand sides of the equations on $\underline{\mathcal{C}}$ follows from Lem. 3.5, due to $\underline{\mathcal{C}} = \underline{\mathcal{C}}((a^* \cdot b^*)^*)$ as $(a^* \cdot b^*)^*$ is the left-hand side of the conclusion. However, we verify the correctness conditions for the left- and the right-hand sides for the (most involved) case of vertex v_1 together as follows (we usually neglect associative brackets, and combine some axiom applications):

$$
\begin{aligned}
(a^* \cdot b^*) \cdot (a^* \cdot b^*)^* &=_{\mathsf{Mil}^-} ((1 + a \cdot a^*) \cdot (1 + b \cdot b^*)) \cdot (a^* \cdot b^*)^* \\
&=_{\mathsf{Mil}^-} (1 \cdot 1 + a \cdot a^* \cdot 1 + 1 \cdot b \cdot b^* + a \cdot a^* \cdot b \cdot b^*) \cdot (a^* \cdot b^*)^* \\
&=_{\mathsf{Mil}^-} (1 + a \cdot a^* + a \cdot a^* \cdot b \cdot b^* + b \cdot b^*) \cdot (a^* \cdot b^*)^* \\
&=_{\mathsf{Mil}^-} (1 + a \cdot a^* \cdot (1 + b \cdot b^*) + b \cdot b^*) \cdot (a^* \cdot b^*)^* \\
&=_{\mathsf{Mil}^-} (1 + a \cdot a^* \cdot b^* + b \cdot b^*) \cdot (a^* \cdot b^*)^* \\
&=_{\mathsf{Mil}^-} 1 \cdot (a^* \cdot b^*)^* + a \cdot (((1 \cdot a^*) \cdot b^*) \cdot (a^* \cdot b^*)^*) + b \cdot ((1 \cdot b^*) \cdot (a^* \cdot b^*)^*) \\
(a+b)^* &=_{\mathsf{Mil}^-} (a+b)^* + (a+b)^* =_{\mathsf{Mil}^-} 1 + (a+b) \cdot (a+b)^* + 1 + (a+b) \cdot (a+b)^* \\
&=_{\mathsf{Mil}^-} 1 + 1 + (a+b) \cdot (a+b)^* + a \cdot (a+b)^* + b \cdot (a+b)^* \\
&=_{\mathsf{Mil}^-} 1 + (a+b) \cdot (a+b)^* + a \cdot (1 \cdot (a+b)^*) + b \cdot (1 \cdot (a+b)^*) \\
&=_{\mathsf{Mil}^-} 1 \cdot (a+b)^* + a \cdot (1 \cdot (a+b)^*) + b \cdot (1 \cdot (a+b)^*)
\end{aligned}
$$

The solution conditions at the vertices v and v_2 can be verified analogously. At v_{11} and at v_{21} the solution conditions follow by uses of the axiom $\mathrm{id}_1(\cdot)$ of Mil^-.

▶ **Lemma 4.3.** *Let* $R \in \{ \overset{\text{coind}}{=\!=\!=}_{\mathcal{S}}, \overset{\text{LLEE}}{=\!=\!=}_{\mathcal{S}} \}$ *for some* $\mathcal{EL}$*-based proof system* $\mathcal{S}$ *with* $\mathsf{ACI} \subseteq \mathcal{S}$. *Then* R *is reflexive, symmetric, and satisfies* $=_{\mathcal{S}} \circ R \subseteq R$, $R \circ =_{\mathcal{S}} \subseteq R$, *and* $=_{\mathcal{S}} \subseteq R$.

▶ **Remark 4.4.** For every coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$, whether $\mathcal{CP}$ is LLEE-witnessed or not, over an $\mathcal{EL}$-based proof system $\mathcal{S}$ with $\mathsf{ACI} \subseteq \mathcal{S} \precsim \mathsf{Mil}$ the finite relation defined by:

$$B := \left\{ \left\langle \tau_{\underline{\mathcal{C}}}(v) + \sum_{i=1}^{n} \underline{a}_i \cdot L_1(v_i),\ \tau_{\underline{\mathcal{C}}}(v) + \sum_{i=1}^{n} \underline{a}_i \cdot L_2(v_i) \right\rangle \;\middle|\; \begin{array}{l} T_{\underline{\mathcal{C}}}(v) = \{ v \xrightarrow{\underline{a}_i} v_i \mid i \in \{1, \ldots, n\} \}, \\ v \in V(\underline{\mathcal{C}}), \text{ the set of vertices of } \underline{\mathcal{C}} \end{array} \right\}$$

is a bisimulation up to $=_{\mathcal{S}}$ with respect to the labeled transition system on all star expressions that is defined by the TSS in Def. 2.1. This can be shown by using that the left-hand sides $L_1(v)$, and respectively the right-hand sides $L_2(v)$, of the equations $L(v)$ in $\mathcal{CP}$, for $v \in V(\underline{\mathcal{C}})$, form $\mathcal{S}$-provable solutions of the 1-chart $\underline{\mathcal{C}}$ that underlies $\mathcal{CP}$.

▶ **Definition 4.5** (proof systems CLC, CC for combining (LLEE-witnessed) coinductive proofs)**.** By the *proof system* $\mathsf{CLC}(A)$ *for combining* LLEE*-witnessed coinductive proofs (over extensions of* $\mathsf{Mil}^-(A)$*) between star expressions over* A we mean the Hilbert-style proof system whose *formulas* are equations between star expressions in $Eq(A)$ or LLEE-witnessed coinductive proofs over $\mathsf{Mil}^-(A)+\Delta$, where $\Delta \in Eq(A)$, and whose *rules* are those of the scheme:

$$\frac{g_1 = h_1 \ \ldots \ g_n = h_n \quad \mathcal{LCP}_{\mathsf{Mil}^-+\Gamma}(e = f)}{e = f}\ \mathrm{LCoProof}_n \quad \begin{array}{c}(\text{where } \Gamma = \{g_1 = h_1, \ldots, g_n = h_n\}, \text{ and} \\ \mathcal{LCP}_{\mathsf{Mil}^-+\Gamma}(e = f) \text{ is a LLEE-witnessed} \\ \text{coind. proof of } e = f \text{ over } \mathsf{Mil}^-+\Gamma)\end{array}$$

where $n \in \mathbb{N}$ (including $n = 0$), and the $(n + 1)$-th premise of an instance of $\mathrm{LCoProof}_n$ consists of a LLEE-witnessed coinductive proof $\mathcal{CP}$ of $e = f$ over Mil^- plus the formulas of the other premises. By the *proof system* $\mathsf{CC}(A)$ *for combining coinductive proofs (over extensions of* $\mathsf{Mil}^-(A)$*) between star expressions over* A we mean the analogous system with a rule $\mathrm{CoProof}_n$ whose $(n + 1)$-th premise is a coinductive proof $\mathcal{CP}$ of $e = f$ over $\mathsf{Mil}^-(A)$ plus the formulas of the other premises that establishes $e \overset{\text{coind}}{=\!=}_{(\mathsf{Mil}^-+\Gamma)} f$ (thus here coinductive proofs do not need to be LLEE-witnessed). Keeping A implicit, we write CLC and CC for $\mathsf{CLC}(A)$ and $\mathsf{CC}(A)$, respectively. Note that CLC and CC do not contain the rules of $\mathcal{EL}$ nor any axioms; instead, derivations have to start with 0-premise instances of $\mathrm{LCoProof}_0$ or $\mathrm{CoProof}_0$.

We now define a coinductively motivated variant cMil of Milner's proof system Mil. In order to obtain cMil we drop the fixed-point rule RSP* from Mil, obtaining Mil^-, and then add a rule each of whose instances use, as a premise, an entire LLEE-witnessed coinductive proof over Mil^- and equations in other premises.

▶ **Definition 4.6** (proof systems cMil, cMil_1, $\overline{\mathsf{cMil}}$)**.** Let A be a set of actions.

The proof system $\mathsf{cMil}(A)$, the *coinductive variant of* $\mathsf{Mil}(A)$, has the same *formulas* as $\mathsf{CLC}(A)$ (formal equations and coinductive proofs), its *axioms* are those of $\mathsf{Mil}^-(A)$, and its *rules* are those of $\mathcal{EL}(A)$, plus the *rule scheme* $\{\mathrm{LCoProof}_n\}_{n\in\mathbb{N}}$ from $\mathsf{CLC}(A)$. By $\mathsf{cMil}_1(A)$ we mean *the simple coinductive variant of* $\mathsf{Mil}(A)$, in which only the rule $\mathrm{LCoProof}_1$ of $\mathsf{CLC}(A)$ is added to the rules and axioms of $\mathsf{Mil}^-(A)$. By $\overline{\mathsf{cMil}}(A)$ we mean the variant of $\mathsf{cMil}(A)$ in which the more general *rule scheme* $\{\mathrm{CoProof}_n\}_{n\in\mathbb{N}}$ from $\mathsf{CC}(A)$ is used instead.

We again permit to write cMil, cMil_1, $\overline{\mathsf{cMil}}$ for $\mathsf{cMil}(A)$, $\mathsf{cMil}_1(A)$, and $\overline{\mathsf{cMil}}(A)$, respectively.

▶ **Lemma 4.7.** *The following theorem subsumption and equivalence statements hold:*

(i) $\mathsf{cMil}_1 \precsim \mathsf{cMil}$.

(ii) $\mathsf{CLC} \sim \mathsf{cMil}$.

(iii) $\mathsf{CC} \sim \overline{\mathsf{cMil}}$.

Proof. Statement (i) is due to $\mathsf{cMil}_1 \subseteq \mathsf{cMil}$, as cMil_1 is a subsystem of cMil that arises by restricting the rule scheme $\{\mathrm{LCoProof}_i\}_{i\in\mathbb{N}}$ to the single rule $\mathrm{LCoProof}_1$.

For (ii), $\mathsf{CLC} \precsim \mathsf{cMil}$ follows from $\mathsf{CLC} \subseteq \mathsf{cMil}$. For showing the converse implication, $\mathsf{CLC} \succsim \mathsf{cMil}$, it suffices to transform an arbitrary derivation in cMil into a derivation in CLC. For this purpose, all instances of axioms and rules of Mil^- have to be eliminated from derivations in cMil, keeping only instances of $\mathrm{LCoProof}_k$ for $k \in \mathbb{N}$. This can be done by extending the equation premises of rules $\mathrm{LCoProof}_k$ if required. For example, a derivation with a bottommost instance of $\mathrm{LCoProof}_1$ in which the subderivation $\mathcal{D}$ does not contain any instances of $\mathrm{LCoProof}_k$ for $k \in \mathbb{N}$ below the conclusions $g_1 = h_1, \ldots, g_m = h_m$ of instances of $\mathrm{LCoProof}_{k_1}, \ldots, \mathrm{LCoProof}_{k_m}$:

$$\dfrac{\dfrac{\mathcal{D}_1 \quad\quad \mathcal{D}_m}{(g_1 = h_1)\ \ldots\ (g_m = h_m)} \quad \dfrac{\mathcal{D}}{g = h} \quad \mathcal{LCP}(e = f)}{e = f}\ \mathrm{LCoProof}_1 \quad\Longrightarrow\quad \dfrac{\mathcal{D}_1 \quad\quad \mathcal{D}_m}{(g_1 = h_1)\ \ldots\ (g_m = h_m) \quad \mathcal{LCP}'(e = f)}{e = f}\ \mathrm{LCoProof}_m$$

can be replaced, on the right, by a single instance of $\mathrm{LCoProof}_m$, where $\mathcal{LCP}'(e = f)$ is, while formally the same as $\mathcal{LCP}(e = f)$, now a LLEE-witnessed coinductive proof over Mil^- plus $g_1 = h_1, \ldots, g_m = h_m$. The latter is possible because the derivation part $\mathcal{D}$ in Mil^- implies that then $g = h$ can be derived from the assumptions in Mil^- as well.

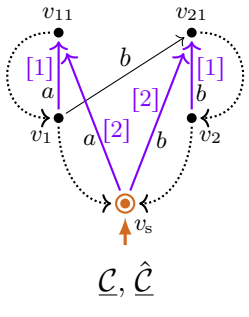
Statement (iii) can be shown entirely analogously as statement (ii). ◀

▶ **Remark 4.8** (completeness of CC, $\overline{\mathsf{cMil}}$, $\overline{\mathsf{Mil}}'$)**.** The proof systems CC and $\overline{\mathsf{cMil}}$, as well as the variant $\overline{\mathsf{Mil}}'$ of Milner's system with the general unique solvability principle USP are complete for bisimilarity of star expressions. This can be established along Salomaa's completeness proof for his inference system for language equality of regular expressions [14], by an argument that we can outline as follows. Given star expressions e and f with $\mathcal{C}(e) \leftrightarrows \mathcal{C}(f)$, e and f can be shown to be principal values of Mil^--provable solutions of $\mathcal{C}(e)$ and $\mathcal{C}(f)$, respectively (by a lemma for the chart interpretation similar to Lem. 3.5). These solutions can be transferred to the (1-free) product chart $\mathcal{C}$ of $\mathcal{C}(e)$ and $\mathcal{C}(f)$, with e and f as principal values of Mil^--provable solutions L_1 and L_2 of $\mathcal{C}$, respectively. From this we obtain a (not necessarily LLEE-witnessed) coinductive proof $\langle \mathcal{C}, L\rangle$ of $e = f$ over Mil^-. It follows that $e = f$ is provable in CC, and in $\overline{\mathsf{cMil}}$. Now since the correctness conditions for the Mil^--provable solutions L_1 and L_2 of $\mathcal{C}$ at each of the vertices of $\mathcal{C}$ together form a guarded system of linear equations to which the rule USP can be applied (as $\mathcal{C}$ is 1-free), we obtain that $e = f$ is also provable in $\overline{\mathsf{Mil}}'$.

5 From LLEE-witnessed coinductive proofs to Milner's system

In this section we show that every LLEE-witnessed coinductive proof over Mil^- of an equation can also be established by a proof in Milner's system Mil. As a consequence we show that the coinductive version cMil of Mil is theorem-subsumed by Mil. We obtain the statements in this section by adapting results in [11, 12, Sect. 5] from LLEE-charts to LLEE-1-charts.

The hierarchical loop structure of a 1-chart $\underline{\hat{\mathcal{C}}}$ with LLEE-witness $\hat{\mathcal{C}}$ facilitates the extraction of a Mil^--provable solution of $\underline{\mathcal{C}}$ (see Lem. 5.4), for the following reason. The behaviour at every vertex w in $\underline{\mathcal{C}}$ can be split into an iteration part that is induced via the loop-entry transitions from w in $\underline{\hat{\mathcal{C}}}$ (which induce loop sub-1-charts with inner loop sub-1-charts whose behaviour can be synthesized recursively), and an exit part that is induced via the body transitions from w in $\underline{\hat{\mathcal{C}}}$. This idea permits to define (Def. 5.1) an "extraction function" $s_{\underline{\hat{\mathcal{C}}}}$ of $\underline{\hat{\mathcal{C}}}$



$$
\begin{aligned}
t_{\hat{\underline{\mathcal{C}}}}(v_{21}, v_2) :=\ & 0^* \cdot (1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_2, v_2)) \equiv 0^* \cdot (1 \cdot 1) =_{\mathsf{Mil}^-} 1 \\
t_{\hat{\underline{\mathcal{C}}}}(v_2, v_s) :=\ & (b \cdot t_{\hat{\underline{\mathcal{C}}}}(v_{21}, v_2))^* \cdot (1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_s, v_s)) =_{\mathsf{Mil}^-} b^* \\
t_{\hat{\underline{\mathcal{C}}}}(v_{21}, v_s) :=\ & 0^* \cdot (1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_2, v_s)) =_{\mathsf{Mil}^-} 1 \cdot b^* =_{\mathsf{Mil}^-} b^* \\
t_{\hat{\underline{\mathcal{C}}}}(v_{11}, v_1) :=\ & 0^* \cdot (1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_1, v_1)) \equiv 0^* \cdot (1 \cdot 1) =_{\mathsf{Mil}^-} 1 \\
t_{\hat{\underline{\mathcal{C}}}}(v_1, v_s) :=\ & (a \cdot t_{\hat{\underline{\mathcal{C}}}}(v_{11}, v_s))^* \cdot (b \cdot t_{\hat{\underline{\mathcal{C}}}}(v_{21}, v_s) + 1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_s, v_s)) \\
=_{\mathsf{Mil}^-}\ & a^* \cdot (b \cdot b^* + 1) =_{\mathsf{Mil}^-} a^* \cdot b^* \\
t_{\hat{\underline{\mathcal{C}}}}(v_{11}, v_s) :=\ & 0^* \cdot (1 \cdot t_{\hat{\underline{\mathcal{C}}}}(v_1, v_s)) =_{\mathsf{Mil}^-} a^* \cdot b^* \\
s_{\hat{\underline{\mathcal{C}}}}(v_s) :=\ & (a \cdot t_{\hat{\underline{\mathcal{C}}}}(v_{11}, v_s) + b \cdot t_{\hat{\underline{\mathcal{C}}}}(v_{21}, v_s))^* \cdot 1 \\
=_{\mathsf{Mil}^-}\ & (a \cdot (a^* \cdot b^*) + b \cdot b^*)^*
\end{aligned}
$$

Figure 2 Extraction of the principal value $s_{\hat{\underline{\mathcal{C}}}}(v_s)$ of a Mil^--provable solution $s_{\hat{\underline{\mathcal{C}}}}$ from the LLEE-witness $\hat{\mathcal{C}}$ used for the coinductive proof in Ex. 4.2. We shorten solution values by using axioms of Mil^-.

from a "relative extraction function" $t_{\hat{\underline{\mathcal{C}}}}$ of $\hat{\underline{\mathcal{C}}}$ whose values $t_{\hat{\underline{\mathcal{C}}}}(w, v)$ capture the behaviour at w in a loop sub-1-chart at v until v is reached. By the same idea the fixed-point rule RSP^* can be used for showing that any two Mil-provable solutions of $\underline{\mathcal{C}}$ are Mil-provably equal (Lem. 5.6). We provide examples for these statements that hint at their proofs (Ex. 5.2, Ex. 5.8), but we refer to the extended version [9] for the detailed arguments that are similar as in [11, 12].

▶ **Definition 5.1** ((relative) extraction function). Let $\underline{\mathcal{C}} = \langle V, A, 1, v_s, \to, \downarrow \rangle$ be a 1-chart with guarded LLEE-witness $\hat{\underline{\mathcal{C}}}$. The *extraction function* $s_{\hat{\underline{\mathcal{C}}}} : V \to \mathit{StExp}(A)$ *of* $\hat{\underline{\mathcal{C}}}$ is defined from the *relative extraction function* $t_{\hat{\underline{\mathcal{C}}}} : \{\langle w, v \rangle \mid w, v \in V(\underline{\mathcal{C}}),\ w \leftharpoonup^{=} v\}$ *of* $\hat{\underline{\mathcal{C}}}$ as follows, for $w, v \in V$:

$$
t_{\hat{\underline{\mathcal{C}}}}(w, v) := \begin{cases} 1 & \text{if } w = v, \\ \Big(\sum_{i=1}^{n} a_i \cdot t_{\hat{\underline{\mathcal{C}}}}(w_i, w)\Big)^* \cdot \Big(\sum_{i=1}^{m} \underline{b}_i \cdot t_{\hat{\underline{\mathcal{C}}}}(u_i, v)\Big) & \text{if } w \leftharpoonup v, \end{cases}
$$

$$
s_{\hat{\underline{\mathcal{C}}}}(w) := \Big(\sum_{i=1}^{n} a_i \cdot t_{\hat{\underline{\mathcal{C}}}}(w_i, w)\Big)^* \cdot \Big(\tau_{\underline{\mathcal{C}}}(w) + \sum_{i=1}^{m} \underline{b}_i \cdot s_{\hat{\underline{\mathcal{C}}}}(u_i)\Big),
$$

given: $T_{\hat{\underline{\mathcal{C}}}}(w) = \big\{w \xrightarrow{a_i}_{[l_i]} w_i \mid l_i \in \mathbb{N}^+,\ i \in \{1, \ldots, n\}\big\} \cup \big\{w \xrightarrow{\underline{b}_i}_{[\mathrm{bo}]} u_i \mid i \in \{1, \ldots, m\}\big\}$,

induction for $t_{\hat{\underline{\mathcal{C}}}}$ on: $\langle w_1, v_1 \rangle <_{\mathrm{lex}} \langle w_2, v_2 \rangle :\iff v_1 \leftharpoonup^{+} v_2 \vee (\, v_1 = v_2 \wedge w_1 \leftarrow^{+}_{\mathrm{bo}} w_2 \,)$,

induction for $s_{\hat{\underline{\mathcal{C}}}}$ on the strict partial order $\leftarrow^{+}_{\mathrm{bo}}$ (see Lem. 3.1),

where $<_{\mathrm{lex}}$ is a well-founded strict partial order due to Lem. 3.1. The choice of the list representations of action-target sets of $\hat{\underline{\mathcal{C}}}$ changes these definition only up to provability in ACI.

▶ **Example 5.2.** We consider the 1-chart $\underline{\mathcal{C}}$, and the LLEE-witness $\hat{\underline{\mathcal{C}}}$ of $\underline{\mathcal{C}}$, in the LLEE-witnessed coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ of $(a^* \cdot b^*)^* = (a + b)^*$ in Ex. 4.2. We detail in Fig. 2 the process of computing the principal value $s_{\hat{\underline{\mathcal{C}}}}(v_s)$ of the extraction function $s_{\hat{\underline{\mathcal{C}}}}$ of $\hat{\underline{\mathcal{C}}}$. The statement of Lem. 5.4 below will guarantee that $s_{\hat{\underline{\mathcal{C}}}}$ is a Mil^--provable solution of $\underline{\mathcal{C}}$.

▶ **Lemma 5.3.** *Let* $\underline{\mathcal{C}}$ *be a weakly guarded* LLEE-1*-chart with guarded* LLEE*-witness* $\hat{\underline{\mathcal{C}}}$. *Then* $s_{\hat{\underline{\mathcal{C}}}}(w) =_{\mathsf{Mil}^-} t_{\hat{\underline{\mathcal{C}}}}(w, v) \cdot s_{\hat{\underline{\mathcal{C}}}}(v)$ *holds for all vertices* $w, v \in V(\underline{\mathcal{C}})$ *such that* $w \leftharpoonup^{=} v$.

▶ **Lemma 5.4** (extracted function is provable solution). *Let $\underline{\mathcal{C}}$ be a w.g.* LLEE*-1-chart with guarded* LLEE*-witness $\hat{\underline{\mathcal{C}}}$. Then the extraction function $s_{\hat{\underline{\mathcal{C}}}}$ of $\hat{\underline{\mathcal{C}}}$ is a* Mil$^-$*-provable solution of $\underline{\mathcal{C}}$.*

▶ **Lemma 5.5.** *Let $\underline{\mathcal{C}}$ be a 1-chart $\underline{\mathcal{C}}$ with guarded* LLEE*-witness $\hat{\underline{\mathcal{C}}}$. Let $\mathcal{S}$ be an $\mathcal{EL}$-based proof system over $StExp(A)$ such that* ACI $\subseteq \mathcal{S} \precsim$ Mil.

Let $s : V(\underline{\mathcal{C}}) \to StExp(A)$ be an $\mathcal{S}$-provable solution of $\underline{\mathcal{C}}$. Then $s(w) =_{\mathsf{Mil}} t_{\hat{\underline{\mathcal{C}}}}(w, v) \cdot s(v)$ holds for all vertices $w, v \in V(\underline{\mathcal{C}})$ with $w \leftsquigarrow^{=} v$.

For an $\mathcal{EL}$-based proof system $\mathcal{S}$ over $StExp(A)$ we say that two star expression functions $s_1, s_2 : V \to StExp(A)$ are *$\mathcal{S}$-provably equal* if $s_1(v) =_{\mathcal{S}} s_2(v)$ holds for all $v \in V$.

▶ **Lemma 5.6** (provable equality of solutions of LLEE-1-charts). *Let $\underline{\mathcal{C}}$ be a guarded* LLEE*-1-chart, and let $\mathcal{S}$ be an $\mathcal{EL}$-based proof system over $StExp(A)$ such that* ACI $\subseteq \mathcal{S} \precsim$ Mil.

Then any two $\mathcal{S}$-provable solutions of $\underline{\mathcal{C}}$ are Mil*-provably equal.*

▶ **Proposition 5.7.** *For every $\mathcal{EL}$-based proof system $\mathcal{S}$ over $StExp(A)$ with* ACI $\subseteq \mathcal{S} \precsim$ Mil*, provability by* LLEE*-witnessed coinductive proofs over $\mathcal{S}$ implies derivability in* Mil*:*

$$\big(e_1 \overset{\text{LLEE}}{=\!=}_{\mathcal{S}} e_2 \quad \Longrightarrow \quad e_1 =_{\mathsf{Mil}} e_2 \big) \qquad \text{for all } e_1, e_2 \in StExp(A). \tag{5.1}$$

Proof. For showing (5.1), let $e, f \in StExp(A)$ be such that $e \overset{\text{LLEE}}{=\!=}_{\mathcal{S}} f$. Then there is a LLEE-witnessed coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ of $e_1 = e_2$ over $\mathcal{S}$. Then $\underline{\mathcal{C}}$ is a LLEE-1-chart, and there are $\mathcal{S}$-provable solutions $L_1, L_2 : V(\underline{\mathcal{C}}) \to StExp(A)$ of $\underline{\mathcal{C}}$ such that $e_1 \equiv L_1(v_{\mathrm{s}})$ and $e_2 \equiv L_2(v_{\mathrm{s}})$. Then L_1 and L_2 are Mil-provably equal by Lem. 5.6. As a consequence we find $e_1 \equiv L_1(v_{\mathrm{s}}) =_{\mathsf{Mil}} L_2(v_{\mathrm{s}}) \equiv e_2$, and hence $e_1 =_{\mathsf{Mil}} e_2$. ◀

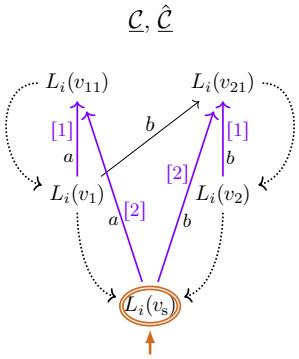
▶ **Example 5.8.** We consider again the LLEE-witnessed coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}, L \rangle$ of $(a^* \cdot b^*)^* = (a + b)^*$ in Ex. 4.2. In Fig. 3 we exhibit the extraction process of derivations in Mil of $L_1(v_{\mathrm{s}}) = s_{\underline{\mathcal{C}}}(v)$ and $L_2(v_{\mathrm{s}}) = s_{\underline{\mathcal{C}}}(v)$ from the LLEE-witness $\hat{\underline{\mathcal{C}}}$ of $\underline{\mathcal{C}}$, which can be combined by $\mathcal{EL}$ rules to obtain a derivation in Mil of $(a^* \cdot b^*)^* \equiv L_1(v_{\mathrm{s}}) = L_2(v_{\mathrm{s}}) \equiv (a + b)^*$.

▶ **Theorem 5.9.** cMil $\precsim$ Mil. *Moreover, every derivation in* cMil *with conclusion $e = f$ can be transformed effectively into a derivation in* Mil *that has the same conclusion.*

Proof. It suffices to show the transformation statement. This can be established by a straightforward induction on the depth of derivations in cMil, in which the only non-trivial case is the elimination of LCoProof$_n$ instances. For every instance of LCoProof$_n$, see Def. 4.1, the induction hypothesis guarantees that the first n premises $g_1 = h_1, \ldots, g_n = h_n$ are derivable in Mil. Then the $(n + 1)$-th premise $\mathcal{LCP}_{\mathsf{Mil}^- + \Gamma}(e = f)$ for $\Gamma := \{g_i = h_i \mid i \in \{1, \ldots, n\}\}$ is also a LLEE-witnessed coinductive proof of $e = f$ over Mil. Therefore we can apply Prop. 5.7 in order to obtain a derivation of $e = f$ in Mil, the conclusion of the LCoProof$_n$ instance. ◀

6 From Milner's system to LLEE-witnessed coinductive proofs

In this section we develop a proof-theoretic interpretation of Mil in cMil$_1$, and hence in cMil. The crucial step hereby is to show that every instance ι of the fixed-point rule RSP* of Mil can be mimicked by a LLEE-witnessed coinductive proof over Mil$^-$ in which also the premise of ι may be used. Specifically, an RSP*-instance with premise $e = f \cdot e + g$ and conclusion $e = f^* \cdot g$ can be translated into a coinductive proof of $e = f^* \cdot g$ over Mil$^-$+$\{e = f \cdot e + g\}$ with underlying 1-chart $\underline{\mathcal{C}}(f^* \cdot g)$ and LLEE-witness $\hat{\underline{\mathcal{C}}}(f^* \cdot g)$. While we have illustrated this transformation already in Fig. 1 in the Introduction, we detail it also for a larger example.



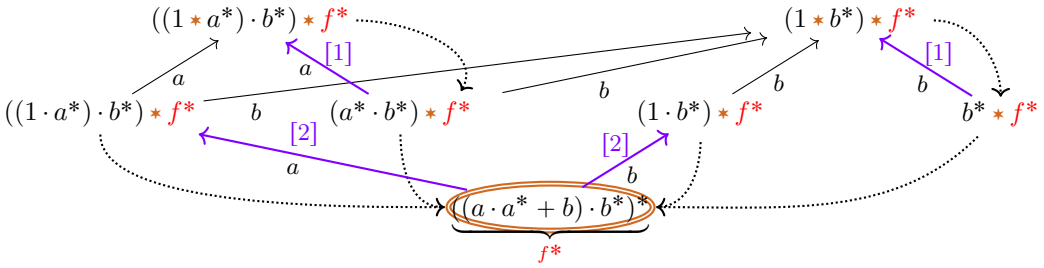
$$L_i(v_{21}) =^{(\mathrm{sol})}_{\mathsf{Mil}^-} 1 \cdot L_i(v_2) =_{\mathsf{Mil}^-} L_i(v_2)$$
$$(=^{(\mathrm{sol})}_{\mathsf{Mil}^-} \text{ means use of 'is } \mathsf{Mil}^-\text{-provable solution'})$$
$$L_i(v_2) =^{(\mathrm{sol})}_{\mathsf{Mil}^-} b \cdot L_i(v_{21}) + 1 \cdot L_i(v_s) =_{\mathsf{Mil}^-} b \cdot L_i(v_2) + L_i(v_s)$$
$$\Downarrow \text{ applying RSP}^*$$
$$L_i(v_2) =_{\mathsf{Mil}} b^* \cdot L_i(v_s)$$
$$L_i(v_{11}) =^{(\mathrm{sol})}_{\mathsf{Mil}^-} 1 \cdot L_i(v_1) =_{\mathsf{Mil}^-} L_i(v_1)$$
$$L_i(v_1) =_{\mathsf{Mil}^-} a \cdot L_i(v_{11}) + b \cdot L_i(v_{21}) + 1 \cdot L_i(v_s)$$
$$=_{\mathsf{Mil}} a \cdot L_i(v_1) + (b \cdot b^* + 1) \cdot L_i(v_s)$$
$$=_{\mathsf{Mil}^-} a \cdot L_i(v_1) + b^* \cdot L_i(v_s)$$
$$\Downarrow \text{ applying RSP}^*$$
$$L_i(v_1) =_{\mathsf{Mil}} a^* \cdot (b^* \cdot L_i(v_s)) =_{\mathsf{Mil}^-} (a^* \cdot b^*) \cdot L_i(v_s)$$
$$L_i(v_s) =^{(\mathrm{sol})}_{\mathsf{Mil}^-} 1 + a \cdot L_i(v_{11}) + b \cdot L_i(v_{21}) =_{\mathsf{Mil}^-} 1 + a \cdot L_i(v_1) + b \cdot L_i(v_2)$$
$$=_{\mathsf{Mil}} (a \cdot (a^* \cdot b^*) + b \cdot b^*) \cdot L_i(v_s) + 1$$
$$\Downarrow \text{ applying RSP}^*$$
$$L_i(v_s) =_{\mathsf{Mil}} (a \cdot (a^* \cdot b^*) + b \cdot b^*)^* \cdot 1 =_{\mathsf{Mil}^-} (a \cdot (a^* \cdot b^*) + b \cdot b^*)^* =_{\mathsf{Mil}^-} s_{\hat{\underline{\mathcal{C}}}}(v_s)$$

Figure 3 Use of the LLEE-witness $\hat{\underline{\mathcal{C}}}$ underlying the coinductive proof $\langle \underline{\mathcal{C}}, L \rangle$ in Ex. 4.2 for showing that the principal value $L_i(v_s)$ of the Mil^--provable solution L_i for $i \in \{1,2\}$ is Mil-provably is Mil-provably equal to the principal value $s_{\hat{\underline{\mathcal{C}}}}(v_s)$ of the solution $s_{\hat{\underline{\mathcal{C}}}}$ extracted from $\hat{\underline{\mathcal{C}}}$ (see Fig. 2).

▶ **Example 6.1.** We consider an instance of RSP* that corresponds, up to an application of r-distr$(+,\cdot)$, to the instance of RSP* at the bottom in Fig. 3:

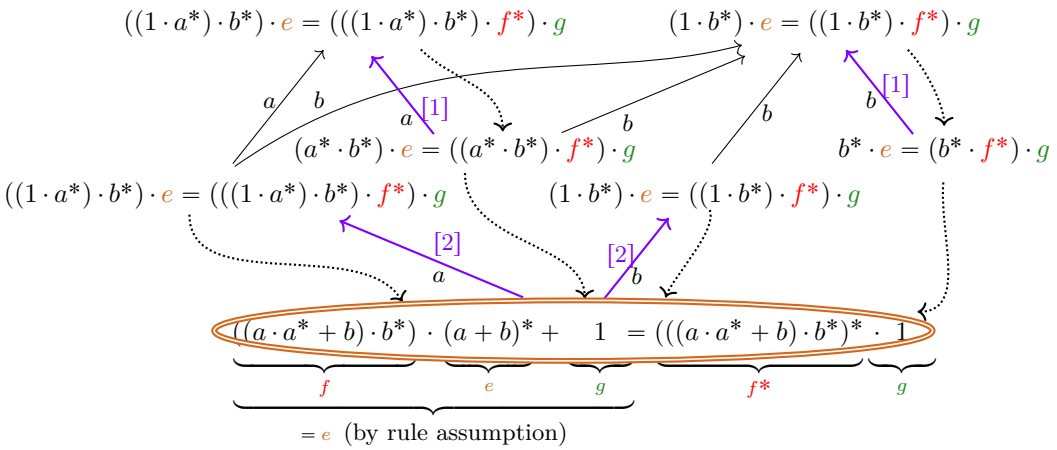
$$\frac{\overbrace{(a+b)^*}^{e} = \overbrace{((a \cdot a^* + b) \cdot b^*)}^{f} \cdot \overbrace{(a+b)^*}^{e} + \overbrace{1}^{g}}{\underbrace{(a+b)^*}_{e} = \underbrace{((a \cdot a^* + b) \cdot b^*)^*}_{f^*} \cdot \underbrace{1}_{g}} \ \mathrm{RSP}^* \quad (\text{where } f\!\downarrow) \tag{6.1}$$

We want to mimic this instance by one of LCoProof$_1$ that uses a LLEE-witnessed coinductive proof of $e = f^* \cdot g$ over Mil^- plus the premise of the RSP* instance. We first obtain the 1-chart interpretation $\underline{\mathcal{C}}(f^*)$ of f^* according to Def. 3.2 together with its LLEE-witness $\hat{\underline{\mathcal{C}}}(f^*)$:



Due to Lem. 3.5 the iterated partial 1-derivatives as depicted define a Mil^--provable solution of $\underline{\mathcal{C}}(f^*)$ when stacked products $\star$ are replaced by products $\cdot$. From this LLEE-witness that carries a Mil-provable solution we now obtain a LLEE-witnessed coinductive proof of $f \cdot e + g = f^* \cdot g$ under the assumption of $e = f \cdot e + g$, as follows. By replacing parts $(\ldots) \star f^*$

by $\pi((\ldots))\cdot e$ in the Mil-provable solution of $\underline{\mathcal{C}}(f^*)$, and respectively, by replacing $(\ldots)\star f^*$ by $(\pi((\ldots))\cdot f^*)\cdot g$ we obtain the left- and the right-hand sides of the formal equations below:



This is a LLEE-witnessed coinductive proof $\mathcal{LCP}$ of $f\cdot e+g = f^*\cdot g$ over $\mathsf{Mil}^-+\{e = f\cdot e+g\}$: The right-hand sides form a Mil-provable solution of $\mathcal{C}(f^*\cdot g)$ due to Lemma 3.5 (note that $\mathcal{C}(f^*\cdot g)$ is isomorphic to $\mathcal{C}(f^*)$ due to $g\equiv 1$). The left-hand sides also form a solution of $\mathcal{C}(f^*\cdot g)$ (see Lem. 6.2 below), noting that for the 1-transitions back to the conclusion the assumption $e = f\cdot e+g$ must be used in addition to Mil^-. By using this assumption again, the result $\mathcal{LCP}'$ of replacing $f\cdot e+g$ in the conclusion of $\mathcal{LCP}$ by e is also a LLEE-witnessed coinductive proof over $\mathsf{Mil}^-+\{e = f\cdot e+g\}$. Consequently:

$$\frac{e = f\cdot e+g \quad \mathcal{LCP}_{\mathsf{Mil}^-+\{e=f\cdot e+g\}}(e = f^*\cdot g)}{e = f^*\cdot g}\ \mathrm{LCoProof}_1$$

is a rule instance of cMil and CLC by which we have mimicked the RSP* instance in (6.1).

▶ **Lemma 6.2.** *Let $e,f,g\in \mathit{StExp}(A)$ with $f{\downarrow}$, and let $\Gamma := \{e = f\cdot e+g\}$. Then e is the principal value of a $(\mathsf{Mil}^-+\Gamma)$-provable solution of the 1-chart interpretation $\underline{\mathcal{C}}(f^*\cdot g)$ of $f^*\cdot g$.*

Proof. First, it can be verified that the vertices of $\underline{\mathcal{C}}(f^*\cdot g)$ are of either of three forms:

$$V(\underline{\mathcal{C}}(f^*\cdot g)) = \{f^*\cdot g\}\cup\big\{(F\star f^*)\cdot g \;\big|\; F\in\underline{\partial}^+(f)\big\}\cup\big\{G \mid G\in\underline{\partial}^+(g)\big\}\,,$$

where $\underline{\partial}^+(f)$ means the set of iterated 1-derivatives of f according to the TSS in Def. 3.2. This facilitates to define a star expression function $s : V(\underline{\mathcal{C}}(f^*\cdot g))\to \mathit{StExp}(A)$ on $\underline{\mathcal{C}}(f^*\cdot g)$ by:

$$s(f^*\cdot g) := e\,,\qquad s((F\star f^*)\cdot g) := \pi(F)\cdot e\,,\qquad s(G) := \pi(G)\,,$$

for all $F\in\underline{\partial}^+(f)$ and $G\in\underline{\partial}^+(g)$. We show that s is a $(\mathsf{Mil}^-+\Gamma)$-provable solution of $\underline{\mathcal{C}}(f^*\cdot g)$.

For this, we have to show that s is $(\mathsf{Mil}^-+\Gamma)$-provable at each of the three kinds of vertices of $\underline{\mathcal{C}}(f^*\cdot g)$, namely at $f^*\cdot g$, at $(F\star f^*)\cdot g$ with $F\in\underline{\partial}^+(f)$, and at G with $G\in\underline{\partial}^+(g)$. Here we only consider $f^*\cdot g$, because it is the single case in which the assumption in Γ has to be used, and the two other forms can be argued similarly (please see in the appendix). By $\underline{A\partial}(H) := \{\langle\underline{a}, H'\rangle \mid H\xrightarrow{\underline{a}} H'\}$ we denote the set of "action 1-derivatives" of a stacked star expression H. In the following argument we avoid list representations of transitions

(as in Def. 2.7) in favor of arguing with sums over sets of action derivatives that represent ACI-equivalence classes of star expressions. This shorthand is possible due to $\mathsf{ACI} \subseteq \mathsf{Mil}^-$.

$$
\begin{aligned}
s(f^* \cdot g) \equiv\ & e && \text{(by the definition of } s\text{)}\\
=_{\mathsf{Mil}^-+\Gamma}\ & f \cdot e + g && \text{(since } \Gamma = \{e = f \cdot e + g\}\text{)}\\
=_{\mathsf{Mil}^-}\ & \Big(\tau_{\underline{\mathcal{C}}(f)}(f) + \sum_{\langle \underline{a}, F\rangle \in \underline{A\partial}(f)} \underline{a} \cdot \pi(F)\Big) \cdot e + \Big(\tau_{\underline{\mathcal{C}}(g)}(g) + \sum_{\langle \underline{a}, G\rangle \in \underline{A\partial}(g)} \underline{a} \cdot \pi(G)\Big) && \text{(by using Lem. 3.4)}\\
=_{\mathsf{Mil}^-}\ & \Big(\sum_{\langle \underline{a}, F\rangle \in \underline{A\partial}(f)} \underline{a} \cdot (\pi(F) \cdot e)\Big) + \Big(\tau_{\underline{\mathcal{C}}(f^* \cdot g)}(f^* \cdot g) + \sum_{\langle \underline{a}, G\rangle \in \underline{A\partial}(g)} \underline{a} \cdot \pi(G)\Big) && \text{(by (assoc}(\cdot)\text{), (r-distr}(+,\cdot)\text{), (deadlock))}\\
& \text{(using } \tau_{\underline{\mathcal{C}}(f)}(f) \equiv 0 \text{ due to } f{\downarrow}\text{, and } \tau_{\underline{\mathcal{C}}(f^* \cdot g)}(f^* \cdot g) \equiv \tau_{\underline{\mathcal{C}}(g)}(g)\text{)}\\
=_{\mathsf{ACI}}\ & \tau_{\underline{\mathcal{C}}(f^* \cdot g)}(f^* \cdot g) + \Big(\sum_{\langle \underline{a}, F\rangle \in \underline{A\partial}(f)} \underline{a} \cdot (s((F * f^*) \cdot g))\Big) + \sum_{\langle \underline{a}, G\rangle \in \underline{A\partial}(g)} \underline{a} \cdot s(G)\\
& \text{(by definition of } s\text{, axioms (comm}(+)\text{))}\\
=_{\mathsf{ACI}}\ & \tau_{\underline{\mathcal{C}}(f^* \cdot g)}(f^* \cdot g) + \sum_{\langle \underline{a}, E'\rangle \in \underline{A\partial}(f^* \cdot g)} \underline{a} \cdot s(E')\\
& \text{(since } \underline{A\partial}(f^* \cdot g) = \{\langle \underline{a}, (F * f^*) \cdot g\rangle \mid \langle \underline{a}, F\rangle \in \underline{A\partial}(f)\} \cup \underline{A\partial}(g)\\
& \text{by inspection of the TSS in Def. 3.2).}
\end{aligned}
$$

Due to $\mathsf{ACI} \subseteq \mathsf{Mil}^- \subseteq \mathsf{Mil}^-+\Gamma$, the above chain of equalities is provable in $\mathsf{Mil}^-+\Gamma$. Therefore it demonstrates, for any given list representation of $T_{\underline{\mathcal{C}}(f^* \cdot g)}(f^* \cdot g)$ according to the correctness condition in Def. 2.7, that s is a $(\mathsf{Mil}^-+\Gamma)$-provable solution of $\underline{\mathcal{C}}(f^* \cdot g)$ at $f^* \cdot g$. ◀

▶ **Lemma 6.3.** *Let $e, f, g \in \mathit{StExp}(A)$ with $f{\downarrow}$, and let $\Gamma := \{e = f \cdot e + g\}$. Then it holds that $e \overset{\text{LLEE}}{=\!=}_{(\mathsf{Mil}^-+\Gamma)} f^* \cdot g$.*

Proof. By Lem. 6.2 there is a $\mathsf{Mil}^-+\Gamma$-provable solution s_1 of $\underline{\mathcal{C}}(f^* \cdot g)$ with $s_1(f^* \cdot g) \equiv e$. By Lem. 3.5 there is a Mil^--provable solution s_2 of $\underline{\mathcal{C}}(f^* \cdot g)$ with $s_2(f^* \cdot g) \equiv f^* \cdot g$. Then $\langle \underline{\mathcal{C}}(f^* \cdot g), L\rangle$ with $L(v) := s_1(v) = s_2(v)$ for all $v \in V(\underline{\mathcal{C}}(f^* \cdot g))$ is a LLEE-witnessed coinductive proof of $e = f^* \cdot g$ over $\mathsf{Mil}^-+\Gamma$, as $\underline{\mathcal{C}}(f^* \cdot g)$ has LLEE-witness $\hat{\underline{\mathcal{C}}}(f^* \cdot g)$ by Thm. 3.3. ◀

▶ **Theorem 6.4.** $\mathsf{Mil} \precsim \mathsf{cMil}_1$. *What is more, every derivation in* Mil *with conclusion $e = f$ can be transformed effectively into a derivation with conclusion $e = f$ in* cMil_1.

Proof. Every derivation $\mathcal{D}$ in Mil can be transformed into a derivation $\mathcal{D}'$ in cMil_1 with the same conclusion as $\mathcal{D}$ by replacing every instance of RSP* in $\mathcal{D}$ by a mimicking derivation in cMil_1 as in the following step, where $f{\downarrow}$ holds as the side-condition of the instance of RSP*:

$$
\frac{e = f \cdot e + g}{e = f^* \cdot g}\ \text{RSP}^* \qquad \Longmapsto \qquad \frac{e = f \cdot e + g \quad \mathcal{LCP}_{\mathsf{Mil}^-+\Gamma}(e = f^* \cdot g)}{e = f^* \cdot g}\ \text{LCoProof}_1
$$

and where $\mathcal{LCP}_{\mathsf{Mil}^-+\Gamma}(e = f^* \cdot g)$ is, for $\Gamma := \{e = f \cdot e + g\}$, a LLEE-witnessed coinductive proof over $\mathsf{Mil}^-+\Gamma$ of $e = f^* \cdot g$ that is guaranteed by Lem. 6.3. ◀

▶ **Theorem 6.5.** $\mathsf{Mil} \sim \mathsf{cMil}_1 \sim \mathsf{cMil} \sim \mathsf{CLC}$, *i.e. these proof systems are theorem-equivalent.*

Proof. Due to $\mathsf{Mil} \precsim \mathsf{cMil}_1 \precsim \mathsf{cMil}\ (\sim \mathsf{CLC}) \precsim \mathsf{Mil}$ by Thm. 6.4, Lem. 4.7, and Thm. 5.9. ◀

7 Conclusion

In order to increase the options for a completeness proof of Milner's system Mil for the process semantics of regular expressions under bisimilarity, we set out to formulate proof systems of equal strength half-way in between Mil and bisimulations between star expressions. Specifically we aimed at characterizing the derivational power that the fixed-point rule RSP* in Mil adds to its purely equational part Mil^-. We based our development on a crucial step from the completeness proof [11] for a tailored restriction of Mil to "1-free" star expressions: guarded linear specifications with the (layered) loop existence and elimination property (L)LEE [7, 11] are uniquely solvable in Mil. We have obtained the following concepts and results:

- As LLEE-witnessed coinductive proof we defined any weakly guarded LLEE-1-chart $\underline{\mathcal{C}}$ whose vertices are labeled by equations between the values of two provable solutions of $\underline{\mathcal{C}}$.
- Based on such proofs, we defined a coinductive version cMil of Milner's system Mil, and as its "kernel" a system CLC for merely combining LLEE-witnessed coinductive proofs.
- Via proof transformations we showed that cMil and CLC are theorem-equivalent to Mil.
- Based on coinductive proofs without LLEE-witnesses, we formulated systems $\overline{\mathsf{cMil}}$ and CC that can be shown to be complete, as can a variant $\overline{\mathsf{Mil}}'$ of Mil with the strong rule USP.

Since the proof systems cMil and CLC are tied to process graphs via the circular deductions they permit, and as they are theorem-equivalent with Mil, they may become natural beachheads for a completeness proof of Milner's system. Indeed, they can be linked to the completeness proof in [11]: it namely guarantees that valid equations between "1-free" star expressions can always be mimicked by derivations in CLC of depth 2. This suggests the following question:

- ▻ Can derivations in CLC (in cMil) always be simplified to some (kind of) normal form that is of bounded depth (resp., of bounded nesting depth of LLEE-witn. coinductive proofs)?

Investigating workable concepts of "normal form" for derivations in CLC or in cMil, by using simplification steps of process graphs with LEE and 1-transitions under 1-bisimilarity as developed for the completeness proof for "1-free" star expressions in [11], is our next goal.

References

1 Roberto M. Amadio and Luca Cardelli. Subtyping Recursive Types. *ACM Trans. Program. Lang. Syst.*, 15(4):575–631, September 1993. doi:10.1145/155183.155231.

2 Valentin Antimirov. Partial Derivatives of Regular Expressions and Finite Automaton Constructions. *Theoretical Computer Science*, 155(2):291–319, 1996. doi:10.1016/0304-3975(95)00182-4.

3 Michael Brandt and Fritz Henglein. Coinductive Axiomatization of Recursive Type Equality and Subtyping. *Fundamenta Informaticae*, 33(4):309–338, December 1998. doi:10.1007/3-540-62688-3_29.

4 Clemens Grabmayer. *Relating Proof Systems for Recursive Types*. PhD thesis, Vrije Universiteit Amsterdam, March 2005. www.phil.uu.nl/~clemens/linkedfiles/proefschrift.pdf.

5 Clemens Grabmayer. Using Proofs by Coinduction to Find "Traditional" Proofs. In José Luiz Fiadeiro, Neal Harman, Markus Roggenbach, and Jan Rutten, editors, *Proceedings of CALCO 2005*, volume 3629 of *LNCS*, pages 175–193. Springer, 2005.

6 Clemens Grabmayer. A Coinductive Axiomatisation of Regular Expressions under Bisimulation. Technical report, University of Nottingham, 2006. Short Contribution to CMCS 2006, March 25-27, 2006, Vienna Institute of Technology, Austria.

7 Clemens Grabmayer. Modeling Terms by Graphs with Structure Constraints (Two Illustrations). In *Proc. TERMGRAPH@FSCD'18*, volume 288, pages 1–13, http://www.eptcs.org/, 2019. EPTCS. doi:10.4204/EPTCS.288.1.

8 Clemens Grabmayer. Structure-Constrained Process Graphs for the Process Semantics of Regular Expressions. Technical report, `arxiv.org`, December 2020. `arXiv:2012.10869`.

9 Clemens Grabmayer. A Coinductive Version of Milner's Proof System for Regular Expressions Modulo Bisimilarity. Technical report, `arxiv.org`, August 2021. Extended version of this paper. `arXiv:2108.13104`.

10 Clemens Grabmayer. Structure-Constrained Process Graphs for the Process Semantics of Regular Expressions. In Patrick Bahr, editor, Proceedings 11th International Workshop on *Computing with Terms and Graphs,* Online, 5th July 2020, volume 334 of *Electronic Proceedings in Theoretical Computer Science*, pages 29–45. Open Publishing Association, 2021. `doi:10.4204/EPTCS.334.3`.

11 Clemens Grabmayer and Wan Fokkink. A Complete Proof System for 1-Free Regular Expressions Modulo Bisimilarity. In *Proceedings of LICS 2020*, New York, NY, 2020. ACM. Extended report see [12].

12 Clemens Grabmayer and Wan Fokkink. A Complete Proof System for 1-Free Regular Expressions Modulo Bisimilarity. Technical report, `arxiv.org`, April 2020. `arXiv:2004.12740`.

13 Robin Milner. A Complete Inference System for a Class of Regular Behaviours. *Journal of Computer and System Sciences*, 28(3):439–466, 1984.

14 Arto Salomaa. Two Complete Axiom Systems for the Algebra of Regular Events. *Journal of the ACM*, 13(1):158–169, 1966. `doi:10.1145/321312.321326`.

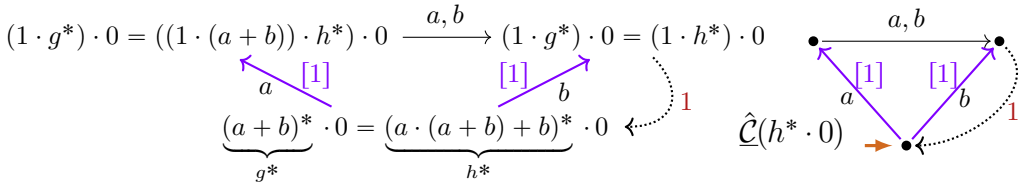
A Appendix: Supplements

This appendix has two parts: In Section A.1, we justify the coinductive proofs we exhibited in Section 1. In Section A.2, we provide remaining details of a lemma that is crucial for the construction of the transformation from the coinductive system cMil to Milner's system Mil.

A.1 Justification of coinductive proofs in Section 1

Here we provide justifications for the coinductive proof on page 2, in Ex. A.1 below, and for the coinductive proof in Fig. 1, in Ex. A.2 below.

▶ **Example A.1** (LLEE-witnessed coinductive proof on page 2)**.** In Section 1 on page 2 we displayed, for the statement $g^* \cdot 0 \equiv (a+b)^* \cdot 0 \overset{\text{LLEE}}{=\!=}_{\mathsf{Mil}^-} (a \cdot (a+b)+b)^* \cdot 0 \equiv h^* \cdot 0$ the coinductive proof $\mathcal{CP} = \langle \underline{\mathcal{C}}(h^* \cdot 0), L \rangle$ over Mil^- with underlying LLEE-witness $\underline{\hat{\mathcal{C}}}(h^* \cdot 0)$, where $\underline{\mathcal{C}}(h^* \cdot 0)$ and $\underline{\hat{\mathcal{C}}}(h^* \cdot 0)$ are defined according to Def. 3.2 and the equation-labeling function L on $\underline{\mathcal{C}}(h^* \cdot 0)$ is defined by the illustration that we repeat here:



The correctness conditions at the start vertex (at the bottom) can be verified as follows:

$$
\begin{aligned}
g^* \cdot 0 \equiv \quad & (a+b)^* \cdot 0 =_{\mathsf{Mil}^-} (1 + (a+b) \cdot (a+b)^*) \cdot 0 =_{\mathsf{Mil}^-} 1 \cdot 0 + ((a+b) \cdot g^*) \cdot 0 \\
=_{\mathsf{Mil}^-} \ & 0 + (a \cdot g^* + b \cdot g^*) \cdot 0 =_{\mathsf{Mil}^-} (a \cdot g^* + b \cdot g^*) \cdot 0 \\
=_{\mathsf{Mil}^-} \ & (a \cdot g^*) \cdot 0 + (b \cdot g^*) \cdot 0 =_{\mathsf{Mil}^-} a \cdot (g^* \cdot 0) + b \cdot (g^* \cdot 0) \\
=_{\mathsf{Mil}^-} \ & a \cdot ((1 \cdot g^*) \cdot 0) + b \cdot ((1 \cdot g^*) \cdot 0)\,, \\
h^* \cdot 0 \equiv \quad & (a \cdot (a+b)+b)^* \cdot 0 =_{\mathsf{Mil}^-} (1 + (a \cdot (a+b)+b) \cdot (a \cdot (a+b)+b)^*) \cdot 0
\end{aligned}
$$

$$
\begin{aligned}
&=_{\mathsf{Mil}^-}\ 1\cdot 0+((a\cdot(a+b)+b)\cdot h^*)\cdot 0 =_{\mathsf{Mil}^-}\ 0+((a\cdot(a+b))\cdot h^*+b\cdot h^*)\cdot 0\\
&=_{\mathsf{Mil}^-}\ (a\cdot((a+b)\cdot h^*)+b\cdot h^*)\cdot 0 =_{\mathsf{Mil}^-}\ ((a\cdot((a+b)\cdot h^*))\cdot 0+(b\cdot h^*)\cdot 0\\
&=_{\mathsf{Mil}^-}\ a\cdot((a+b)\cdot h^*)\cdot 0)+b\cdot(h^*\cdot 0)\\
&=_{\mathsf{Mil}^-}\ a\cdot((1\cdot(a+b))\cdot h^*)\cdot 0)+b\cdot((1\cdot h^*)\cdot 0)\,.
\end{aligned}
$$

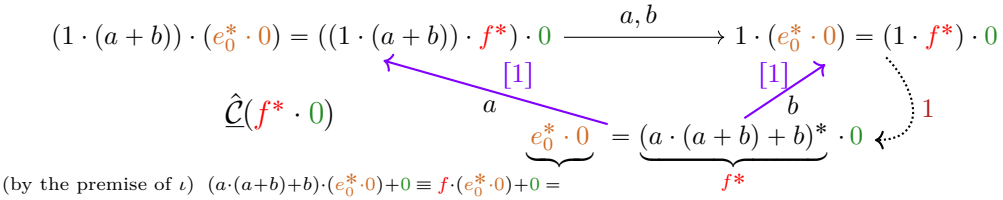
From the provable equality for $g^*\cdot 0$ the correctness condition for $(1\cdot g^*)\cdot 0$ at the left upper vertex of $\underline{\mathcal{C}}(h^*\cdot 0)$ can be obtained by additional uses of the axiom $(\mathrm{id}_1(\cdot))$. The correctness condition for $((1\cdot(a+b))\cdot h^*)\cdot 0$ at the left upper vertex of $\underline{\mathcal{C}}(h^*\cdot 0)$ can be verified as follows:

$$
\begin{aligned}
((1\cdot(a+b))\cdot h^*)\cdot 0 &=_{\mathsf{Mil}^-}\ ((a+b)\cdot h^*)\cdot 0 =_{\mathsf{Mil}^-}\ (a\cdot h^*+b\cdot h^*)\cdot 0\\
&=_{\mathsf{Mil}^-}\ (a\cdot h^*)\cdot 0+(b\cdot h^*)\cdot 0 =_{\mathsf{Mil}^-}\ a\cdot(h^*\cdot 0)+b\cdot(h^*\cdot 0)\\
&=_{\mathsf{Mil}^-}\ a\cdot((1\cdot h^*)\cdot 0)+b\cdot((1\cdot h^*)\cdot 0)\,.
\end{aligned}
$$

Finally, the correctness conditions at the right upper vertex of $\underline{\mathcal{C}}(h^*\cdot 0)$ can be obtained by applications of the axiom $(\mathrm{id}_1(\cdot))$ only.

▶ **Example A.2** (LLEE-witnessed coinductive proof in Fig. 1)**.** We provided a first illustration for translating an instance of the fixed-point rule into a coinductive proof in Figure 1 on page 3. Specifically, we mimicked the instance ι (see below) of the fixed-point rule RSP* in Milner's system $\mathsf{Mil}=\mathsf{Mil}^-{+}\mathrm{RSP}^*$ by a coinductive proof (see also below) over $\mathsf{Mil}^-{+}\{\text{premise of }\iota\}$ with LLEE-witness $\hat{\underline{\mathcal{C}}}(f^*\cdot 0)$:

$$
\frac{\overbrace{(a+b)^*}^{e_0^*}\cdot 0 \;=\; \overbrace{(a\cdot(a+b)+b)}^{f}\cdot(\overbrace{(a+b)^*}^{e_0^*}\cdot 0)+0}{(a+b)^*\cdot 0 \;=\; (a\cdot(a+b)+b)^*\cdot 0}\;\iota,\ \mathrm{RSP}^*\ \frac{e_0^*\cdot 0 \;=\; f\cdot(e_0^*\cdot 0)+0}{e_0^*\cdot 0 \;=\; f^*\cdot 0}
$$



The correctness conditions for the right-hand sides of this prooftree to be a LLEE-witnessed coinductive proof $\mathsf{Mil}^-{+}\{\text{premise of }\iota\}$ are the same as those that we have verified for the right-hand sides of the coinductive proof over Mil^- with the same LLEE-witness in Ex. A.1. Note that the premise of ι is not used for the correctness conditions of the right-hand sides. The correctness condition for the left-hand side $e_0^*\cdot 0$ at the bottom vertex of $\underline{\mathcal{C}}(f^*\cdot 0)$ can be verified as follows, now making use of the premise of the considered instance ι of RSP*:

$$
\begin{aligned}
e_0^*\cdot 0 &=_{\{\text{premise of }\iota\}}\ f\cdot(e_0^*\cdot 0)+0 \ =_{\mathsf{Mil}^-}\ (a\cdot(a+b)+b)\cdot(e_0^*\cdot 0)\\
&=_{\mathsf{Mil}^-}\ (a\cdot(a+b))\cdot(e_0^*\cdot 0)+b\cdot(e_0^*\cdot 0)\\
&=_{\mathsf{Mil}^-}\ a\cdot((a+b)\cdot(e_0^*\cdot 0))+b\cdot(1\cdot(e_0^*\cdot 0))\\
&=_{\mathsf{Mil}^-}\ a\cdot((1\cdot(a+b))\cdot(e_0^*\cdot 0))+b\cdot(1\cdot(e_0^*\cdot 0))
\end{aligned}
$$

Together this yields the provable equation:

$$
e_0^*\cdot 0 \ =_{\mathsf{Mil}^-+\{\text{premise of }\iota\}}\ a\cdot((1\cdot(a+b))\cdot(e_0^*\cdot 0))+b\cdot(1\cdot(e_0^*\cdot 0))\,,
$$

which demonstrates the correctness condition for the left-hand side $e_0^* \cdot 0$ at the bottom vertex of $\underline{\mathcal{C}}(f^* \cdot 0)$. The correctness condition for the left-hand side $a \cdot ((1 \cdot (a + b))$ at the top left vertex of $\underline{\mathcal{C}}(f^* \cdot 0)$ can be verified without using the premise of ι as follows:

$$\begin{aligned}((1 \cdot (a + b)) \cdot (e_0^* \cdot 0) &=_{\mathsf{Mil}^-} ((a + b) \cdot (e_0^* \cdot 0) =_{\mathsf{Mil}^-} a \cdot (e_0^* \cdot 0) + b \cdot (e_0^* \cdot 0) \\ &=_{\mathsf{Mil}^-} a \cdot (1 \cdot (e_0^* \cdot 0)) + b \cdot (1 \cdot (e_0^* \cdot 0))\,.\end{aligned}$$

Finally, the correctness condition of the left-hand side $1 \cdot (e_0^* \cdot 0)$ at the right upper vertex of $\underline{\mathcal{C}}(f^* \cdot 0)$ can be obtained by an application of the axiom $(\mathrm{id}_1(\cdot))$ only.

A.2 Completing the proof of Lemma 6.2

Here we provide those remaining details for the proof of Lem. 6.2 that we have postponed from within the proof environment on page 15. This lemma is crucial for the construction of the proof transformation from the coinductive system cMil to Milner's system Mil (see Thm. 6.4), which proceeds by mimicking arbitrary instances ι of the fixed-point rule RSP* by coinductive proofs over the equational part Mil^- of Mil plus the premise of ι (see Lem. 6.3). Indeed, Lem. 6.2 provides, for every generic instance ι of RSP* as in Def. 2.4, a provable solution for the 1-chart interpretation $\underline{\mathcal{C}}(f^* \cdot g)$ of $f^* \cdot g$ that can provide (see Lem. 6.3) the left-hand sides of the equations in a coinductive proof that mimics ι.

▶ **Lemma** (= Lem. 6.2). *Let $e, f, g \in \mathit{StExp}(A)$ with $f{\downarrow}$, and let $\Gamma := \{e = f \cdot e + g\}$. Then the star expression e is the principal value of a $(\mathsf{Mil}^- + \Gamma)$-provable solution of the 1-chart interpretation $\underline{\mathcal{C}}(f^* \cdot g)$ of $f^* \cdot g$.*

Before we extend the proof of this lemma from page 15 by treating the cases that we have not yet treated, we gather and outline the proofs of auxiliary statements that are used in the proof of Lem. 6.2. The first lemma concerns the set of *action 1-derivatives of* a star expression, or stacked star expression, over set A of actions, defined by $\underline{A\partial}(E) := \{\langle \underline{a}, E' \rangle \mid E \xrightarrow{a} E'\}$, where the transitions are defined by the TSS in Def. 3.2.

▶ **Lemma A.3.** *The action 1-derivatives $\underline{A\partial}(E)$ of a stacked star expression E over actions in A satisfy the following recursive equations, for all $a \in A$, $e, e_1, e_2 \in \mathit{StExp}(A)$, and stacked star expressions E_1 over actions in A:*

$$\begin{aligned}\underline{A\partial}(0) &:= \underline{A\partial}(1) := \varnothing\,,\\ \underline{A\partial}(a) &:= \{\langle a, 1\rangle\}\,,\\ \underline{A\partial}(e_1 + e_2) &:= \underline{A\partial}(e_1) \cup \underline{A\partial}(e_2)\,,\\ \underline{A\partial}(E_1 \cdot e_2) &:= \begin{cases} \{\langle \underline{a}, E_1' \cdot e_2\rangle \mid \langle \underline{a}, E_1'\rangle \in \underline{A\partial}(E_1)\} & \text{if } E_1{\downarrow}, \\ \{\langle \underline{a}, E_1' \cdot e_2\rangle \mid \langle \underline{a}, E_1'\rangle \in \underline{A\partial}(E_1)\} \cup \underline{A\partial}(E_2) & \text{if } E_1{\downarrow}, \end{cases}\\ \underline{A\partial}(E_1 \star e_2^*) &:= \begin{cases} \{\langle \underline{a}, E_1' \cdot e_2^*\rangle \mid \langle \underline{a}, E_1'\rangle \in \underline{A\partial}(E_1)\} & \text{if } E_1{\downarrow}, \\ \{\langle \underline{a}, E_1' \cdot e_2^*\rangle \mid \langle \underline{a}, E_1'\rangle \in \underline{A\partial}(E_1)\} \cup \{\langle 1, e_2^*\rangle\} & \text{if } E_1{\downarrow}, \end{cases}\\ \underline{A\partial}(e^*) &:= \{\langle a, E' \star e^*\rangle \mid \langle a, E'\rangle \in \underline{A\partial}(e)\}\,. \end{aligned} \tag{A.1}$$

Proof. By case-wise inspection of the definition of the TSS in Def. 3.2. ◀

For the proof of Lem. 3.4 we will need the following auxiliary statement.

▶ **Lemma A.4.** *If $e{\downarrow}$ for a star expression $e \in \mathit{StExp}(A)$, then there is a star expression $f \in \mathit{StExp}(A)$ with $f{\downarrow}$, $e =_{\mathsf{Mil}^-} 1 + f$, $|f|_* = |e|_*$, and $((\mathrm{id} \times \pi) \circ \underline{A\partial})(f) = ((\mathrm{id} \times \pi) \circ \underline{A\partial})(e)$.*

Proof. By a proof by induction on structure of e, in which all axioms of Mil^- are used. ◀

▶ **Lemma A.5** (corresponds to Lem. 3.4). *$\pi(E) =_{\mathsf{Mil}^-} \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(E)} \underline{a} \cdot \pi(E')$, for all stacked star expressions E over actions in A, (where we permit that the sum expression on the right indicates a star expression only up to ACI (note that $\mathsf{ACI} \subseteq \mathsf{Mil}^-$)).*

Proof. The statement of the lemma can be proved by induction on the structure of the stacked star expression E with a subinduction on the syntactical star height $|E|_*$ of E. All cases of stacked star expressions can be dealt with in a straightforward manner, except for those with an outermost iteration where in a subcase the subinduction hypothesis must be used.

Suppose that $E \equiv e^*$, and that $e{\downarrow}$ holds. Then by Lem. A.4 there exists a star expression f such that $f{\downarrow}$, $e =_{\mathsf{Mil}} 1+f$, $|f|_* = |e|_*$, and $((\mathrm{id} \times \pi) \circ \underline{A\partial})(f) = ((\mathrm{id} \times \pi) \circ \underline{A\partial})(e)$. Then we can argue as follows to prove the desired Mil^--provable equation for $\pi(E)$, where we apply the subinduction hypothesis for $\pi(f)$, which is possible due to $|f|_* = |e|_* < 1+|e|_* = |e^*|_* = |E|_*$:

$$\begin{aligned}
\pi(E) \equiv\ & \pi(e^*) \equiv e^* =_{\mathsf{Mil}^-} (1+f)^* =_{\mathsf{Mil}^-} f^* =_{\mathsf{Mil}^-} 1 + f \cdot f^* \\
=_{\mathsf{Mil}^-}\ & 1 + f \cdot (1+f)^* =_{\mathsf{Mil}^-} 1 + f \cdot e^* \equiv 1 + \pi(f) \cdot e^* \\
=_{\mathsf{Mil}^-}\ & 1 + \big(\tau_{\underline{\mathcal{C}}(f)}(f) + \textstyle\sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(f)} \underline{a} \cdot \pi(F')\big) \cdot e^* \\
\equiv\ & 1 + \big(0 + \textstyle\sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(f)} \underline{a} \cdot \pi(F')\big) \cdot e^* \\
=_{\mathsf{Mil}^-}\ & \tau_{\underline{\mathcal{C}}(e^*)}(e^*) + \textstyle\sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(e)} \underline{a} \cdot (\pi(E') \cdot e^*) \\
=_{\mathsf{Mil}^-}\ & \tau_{\underline{\mathcal{C}}(e^*)}(e^*) + \textstyle\sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(e)} \underline{a} \cdot \pi(E' * e^*) \\
=_{\mathsf{Mil}^-}\ & \tau_{\underline{\mathcal{C}}(E)}(E) + \textstyle\sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(E)} \underline{a} \cdot \pi(E') \,,
\end{aligned}$$

where in the last step we used the representation of $\underline{A\partial}(E) = \underline{A\partial}(e^*)$ according to (A.1).

In the case that $E = e^*$ with $e{\downarrow}$ we can reason similarly but simpler, because then it is sufficient to use the induction hypothesis for e, which is structurally simpler than e^*. ◀

▶ **Lemma** (= Lem. 3.5). *For every star expression $e \in \mathit{StExp}(A)$ with 1-chart interpretation $\underline{\mathcal{C}}(e) = \langle V(e), A, 1, e, \to, {\downarrow} \rangle$ the star-expression function $s : V(e) \to \mathit{StExp}(A)$, $E \mapsto \pi(E)$ is a Mil^--provably solution of $\underline{\mathcal{C}}(e)$ with principal value e.*

Proof. Immediate consequence of Lem. 3.4. ◀

▶ **Lemma** (= Lem. 6.2). *Let $e, f, g \in \mathit{StExp}(A)$ with $f{\downarrow}$, and let $\Gamma := \{e = f \cdot e + g\}$. Then the star expression e is the principal value of a $(\mathsf{Mil}^-{+}\Gamma)$-provable solution of the 1-chart interpretation $\underline{\mathcal{C}}(f^* \cdot g)$ of $f^* \cdot g$.*

Proof of Lemma 6.2 (extension of the proof on p. 15). First, it can be verified that the vertices of $\underline{\mathcal{C}}(f^* \cdot g)$ are of either of three forms:

$$V(\underline{\mathcal{C}}(f^* \cdot g)) = \{f^* \cdot g\} \cup \big\{(F * f^*) \cdot g \;\big|\; F \in \underline{\partial}^+(f)\big\} \cup \big\{G \mid G \in \underline{\partial}^+(g)\big\} \,, \tag{A.2}$$

where $\underline{\partial}^+(f)$ means the set of iterated 1-derivatives of f according to the TSS in Def. 3.2.

This facilitates to define a function $s : V(\underline{\mathcal{C}}(f^* \cdot g)) \to \mathit{StExp}(A)$ on $\underline{\mathcal{C}}(f^* \cdot g)$ by:

$$\begin{aligned}
s(f^* \cdot g) &:= e \,, \\
s((F * f^*) \cdot g) &:= \pi(F) \cdot e \,, \qquad (\text{for } F \in \underline{\partial}^+(f)), \\
s(G) &:= \pi(G) \qquad (\text{for } G \in \underline{\partial}^+(g)),
\end{aligned}$$

We will show that s is a $(\mathsf{Mil}^-{+}\Gamma)$-provable solution of $\underline{\mathcal{C}}(f^* \cdot g)$. Instead of verifying the correctness conditions for s for list representations of transitions, we will argue more loosely

with sums over action 1-derivatives sets $\underline{A\partial}(H)$ of stacked star expressions H where such sums are only well-defined up to ACI. Due to $\mathsf{ACI} \subseteq \mathsf{Mil}^-$ such an argumentation is possible. Specifically we will demonstrate, for all $E \in V(\underline{\mathcal{C}}(f^* \cdot g))$, that s is a $(\mathsf{Mil}^- + \Gamma)$-provable solution at E, that is, that it holds:

$$s(E) =_{(\mathsf{Mil}^- + \Gamma)} \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(E)} \underline{a} \cdot s(E') \,, \tag{A.3}$$

where by the sum on the right-hand side we mean an arbitrary representative of the ACI equivalence class of star expressions that can be obtained by the sum expression of this form.

For showing (A.3), we distinguish the three cases of vertices $E \in V(\underline{\mathcal{C}}(f^* \cdot g))$ according to (A.2), that is, $E \equiv f^* \cdot g$, $E \equiv (F * f^*) \cdot g$ for some $F \in \underline{\partial}^+(f)$, and $E \equiv G$ for some $G \in \underline{\partial}^+(g)$.

In the first case, $E \equiv f^* \cdot g$, we have argued on page 16 that s is a $(\mathsf{Mil}^- + \Gamma)$-provable solution at E, and hence that (A.3) holds for E as chosen here.

In the second case we consider $E \equiv (F * f^*) \cdot g \in V(\underline{\mathcal{C}}(f^* \cdot g))$. Then $\tau_{\underline{\mathcal{C}}(E)}(E) \equiv \tau_{\underline{\mathcal{C}}((F * f^*) \cdot g)}((F * f^*) \cdot g) \equiv 0$ holds, because expressions with stacked products occurring do not have immediate termination by Def. 3.2. We distinguish the subcases $F{\downarrow}$ and $F{\not\downarrow}$.

For the first subcase we assume $F{\downarrow}$. Then $\tau_{\underline{\mathcal{C}}(F)}(F) \equiv 0$ holds, and we find by Lem. A.3 (or by inspecting the TSS in Def. 3.2):

$$\underline{A\partial}((F * f^*) \cdot g) = \left\{ \langle \underline{a}, (F' * f^*) \cdot g \rangle \mid \langle \underline{a}, F' \rangle \in \underline{A\partial}(F) \right\} . \tag{A.4}$$

Now we argue as follows:

$$\begin{aligned}
s(E) \equiv\ & s((F * f^*) \cdot g) && \text{(in this case)} \\
\equiv\ & \pi(F) \cdot e && \text{(by the definition of } s\text{)} \\
=_{\mathsf{Mil}^-}\ & \Big(\tau_{\underline{\mathcal{C}}(F)}(F) + \sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(F)} \underline{a} \cdot \pi(F') \Big) \cdot e && \text{(by Lem. A.5)} \\
=_{\mathsf{Mil}^-}\ & 0 \cdot e + \sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(F)} \underline{a} \cdot (\pi(F') \cdot e) && \text{(by } \tau_{\underline{\mathcal{C}}(F)}(F) \equiv 0\text{, due to } F{\downarrow}\text{, and axioms (r-distr}(+,\cdot)\text{), (assoc}(\cdot)\text{))} \\
=_{\mathsf{Mil}^-}\ & 0 + \sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(F)} \underline{a} \cdot s((F' * f^*) \cdot g) && \text{(by ax. (deadlock) and def. of } s\text{)} \\
=_{\mathsf{ACI}}\ & \tau_{\underline{\mathcal{C}}((F * f^*) \cdot g)}((F * f^*) \cdot g) + \sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}((F * f^*) \cdot g)} \underline{a} \cdot s(E') && \text{(due to (A.4), and } \tau_{\underline{\mathcal{C}}(E)}(E) \equiv 0\text{)} \\
\equiv\ & \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{\langle \underline{a}, E' \rangle \in \underline{A\partial}(E)} \underline{a} \cdot s(E') && \text{(in this case).}
\end{aligned}$$

For the second subcase we assume $F{\downarrow}$. Then $F \in \mathit{StExp}(A)$ (that is, F does not contain a stacked product symbol), and $\tau_{\underline{\mathcal{C}}(F)}(F) \equiv 1$ holds. Furthermore, we find, again by inspecting the TSS in Def. 3.2:

$$\underline{A\partial}((F * f^*) \cdot g) = \{ \langle 1, f^* \cdot g \rangle \} \cup \left\{ \langle \underline{a}, (F' * f^*) \cdot g \rangle \mid \langle \underline{a}, F' \rangle \in \underline{A\partial}(F) \right\} . \tag{A.5}$$

Now we argue as follows:

$$\begin{aligned}
s(E) \equiv\ & s((F * f^*) \cdot e) && \text{(in this case)} \\
\equiv\ & \pi(F) \cdot e && \text{(by the definition of } s\text{)} \\
=_{\mathsf{Mil}^-}\ & \Big(\tau_{\underline{\mathcal{C}}(F)}(F) + \sum_{\langle \underline{a}, F' \rangle \in \underline{A\partial}(F)} \underline{a} \cdot \pi(F') \Big) \cdot e && \text{(by Lem. A.5)}
\end{aligned}$$

$$
\begin{aligned}
&=_{\mathsf{Mil}^-} && 1\cdot e + \sum_{\langle \underline{a}, F'\rangle \in \underline{A\partial}(f)} \underline{a}\cdot(\pi(F')\cdot e) && \text{(by } \tau_{\underline{\mathcal{C}}(F)}(F)\equiv 1\text{, and axioms (r-distr}(+,\cdot)\text{), (assoc}(\cdot)\text{))}\\
&\equiv && 1\cdot s(f^*\cdot g) + \sum_{\langle \underline{a}, F'\rangle \in \underline{A\partial}(f)} \underline{a}\cdot s((F'\ast f^*)\cdot g) && \text{(by the definition of } s\text{)}\\
&=_{\mathsf{ACI}} && 0 + \sum_{\langle \underline{a}, E'\rangle \in \underline{A\partial}((F\ast f^*)\cdot g)} \underline{a}\cdot s(E') && \text{(by (A.5), using axioms (comm}(+)\text{), and (assoc}(+)\text{))}\\
&\equiv && \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{\langle \underline{a}, E'\rangle \in \underline{A\partial}(E)} \underline{a}\cdot s(E') && \text{(in this case, due to } \tau_{\underline{\mathcal{C}}(E)}(E)\equiv 0\text{).}
\end{aligned}
$$

Due to $\mathsf{ACI} \subseteq \mathsf{Mil}^- \subseteq \mathsf{Mil}^-{+}\Gamma$, the chains of equalities in both subcases are provable in $\mathsf{Mil}^-{+}\Gamma$, and therefore we have verified (A.3) also in the second case.

In the final case, $E = G$ for some $G \in \underline{\partial}^+(g)$, we argue as follows:

$$
\begin{aligned}
s(E) &\equiv && s(G) && \text{(in this case)}\\
&\equiv && \pi(G) && \text{(by the definition of } s\text{)}\\
&=_{\mathsf{Mil}^-} && \tau_{\underline{\mathcal{C}}(G)}(G) + \sum_{\langle \underline{a}, G'\rangle \in \underline{A\partial}(G)} \underline{a}\cdot\pi(G') && \text{(by Lem. A.5)}\\
&=_{\mathsf{ACI}} && \tau_{\underline{\mathcal{C}}(G)}(G) + \sum_{\langle \underline{a}, G'\rangle \in \underline{A\partial}(G)} \underline{a}\cdot s(G') && \text{(by the definition of } s\text{)}\\
&=_{\mathsf{ACI}} && \tau_{\underline{\mathcal{C}}(E)}(E) + \sum_{\langle \underline{a}, E'\rangle \in \underline{A\partial}(E)} \underline{a}\cdot s(E') && \text{(in this case).}
\end{aligned}
$$

Due to $\mathsf{ACI} \subseteq \mathsf{Mil}^- \subseteq \mathsf{Mil}^-{+}\Gamma$, this chain of equalities verifies (A.3) also in this case.

By having established (A.3) for the, according to (A.2), three possible forms of stacked star expressions that are vertices of $\underline{\mathcal{C}}(f^*\cdot g)$, we have established that s is indeed a $(\mathsf{Mil}^-{+}\Gamma)$-provable solution of $\underline{\mathcal{C}}(f^*\cdot g)$. ◀

Functorial Semantics as a Unifying Perspective on Logic Programming

Tao Gu
University College London, UK

Fabio Zanasi
University College London, UK

Abstract

Logic programming and its variations are widely used for formal reasoning in various areas of Computer Science, most notably Artificial Intelligence. In this paper we develop a systematic and unifying perspective for (ground) classical, probabilistic, weighted logic programs, based on categorical algebra. Our departure point is a formal distinction between the syntax and the semantics of programs, now regarded as separate categories. Then, we are able to characterise the various variants of logic program as different *models* for the same syntax category, i.e. structure-preserving functors in the spirit of Lawvere's functorial semantics.

As a first consequence of our approach, we showcase a series of semantic constructs for logic programming pictorially as certain string diagrams in the syntax category. Secondly, we describe the correspondence between probabilistic logic programs and Bayesian networks in terms of the associated models. Our analysis reveals that the correspondence can be phrased in purely syntactical terms, without resorting to the probabilistic domain of interpretation.

2012 ACM Subject Classification Theory of computation → Categorical semantics

Keywords and phrases string diagrams, functorial semantics, logic programming

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.17

1 Introduction

Logic programming is a programming paradigm widely used for knowledge representation in Artificial Intelligence and related fields. In "classical" logic programming, at the basis of formalisms such as Prolog [8] and Datalog [9], programs are sets of Horn clauses, of the form $A \leftarrow B_1, B_2, \ldots, B_n$. However, in the last two decades, various extensions and variants of logic programming emerged in order to handle non-classical reasoning, in which clauses may be associated with a probability or a weight, resulting in a different semantics. These approaches include probabilistic logic programming ([35], Problog [13], PRISM [36], CP-logic [31], PASP [12]) and weighted logic programming [15]. Seemingly different formalisms, such as Bayesian networks, also turn out to be closely related [31].

The main goal of this work is to develop a systematic, unifying framework for the different families of logic programming languages, in which their similarities and differences can be analysed algebraically using the abstract perspective of *category theory*.

Our approach is based on a simple, yet fruitful insight: a formal distinction between the *syntax* and the *semantics* of logic programs. This "separation of concerns" has the benefit of clearly isolating the purely inferential structure underlying a program (syntax) from its "type" (classical, probabilistic, or weighted, expressed by the semantics). Our guiding principle is the perspective of *functorial semantics*, as pioneered by Lawvere [29], in which one encodes an algebraic theory as a category Syn of "syntactic" morphisms (tuples of terms), and then studies models of the theory as *functors* from Syn – the requirement that such functors preserve finite products makes them adhere to the usual notion of model in universal algebra. In analogy, from a logic program $\mathbb{P}$ we will freely generate a category SynLP of

string diagrams [37], capturing the inferential structure of $\mathbb{P}$ as (graphical) syntax. Then, we consider models of SynLP, i.e. structure-preserving functors into other categories, acting as semantic domains of interpretation.

Our main conceptual contribution is the realisation that the different flavours of logic program (classical, probabilistic and weighted) amount to different classes of models of the same syntax category.

This perspective has various consequences. As the syntax is expressed as a freely generated category of string diagrams, not only it includes the clauses of the program as morphisms, but it allows to combine them into more elaborated representations. This provides much flexibility in expressing various kinds of semantic constructs, which may be observed in the image of the models/functors. We use this observation to provide a string diagrammatic description of semantics commonly found in the logic programming literature, such as the immediate consequence operator, the Herbrand semantics, and the stratified semantics, see e.g. [17, 3, 18, 38], as well as the distribution semantics of probabilistic programs, and the standard semantics of weighted programs.

Another payoff of our approach is providing an original perspective on the correspondence between probabilistic logic programs and Bayesian networks. It is a folklore result (*cf.* [31]) that the two formalisms can be translated one into another, modulo some caveats. In the context of our framework, we take advantage of the fact that Bayesian networks are also susceptible of a description in terms of functorial semantics, see [16, 22], and study their correspondence with probabilistic programs in terms of the associated models. Again, the distinction between syntax and semantics provides a valuable insight: it turns out that the correspondence can be entirely expressed at the level of the syntax categories – in contrast with traditional approaches, where it involves a mixture of transformations between graphs, programs, and conditional probabilities. Furthermore, thanks to the use of *string diagrams*, both the combinatorial structure of Bayesian networks (directed acyclic graphs) and the syntax of probabilistic programs can be expressed uniformly as entities of the same kind.

Synopsis. Section 2 is for preliminaries. We introduce the syntax category and present the functorial semantics of classical logic programming in Section 3. We then consider probabilistic logic programming in Section 4, and study the correspondence with Bayesian networks at a functorial level. Section 5 briefly illustrates the case of weighted logic programming. Section 6 is devoted to future work. Missing proofs may be found in Appendix B.

2 Preliminaries

Logic programming. We briefly recall the basics of logic programming, and refer to [33] for more details. Throughout this paper we focus on *ground* logic programming, i.e. in which programs have no variables. A *logic program* (LP) $\mathbb{P}$ based on a set of atoms At is a finite set of clauses φ of the form $A \leftarrow L_1, \dots, L_m.$, where A is an atom and each L_i is a literal (an atom B or a negated atom $\neg B$). The atom A and the set of literals $\{L_1, \dots, L_m\}$ are called the head (denoted as $\mathsf{head}(\varphi)$) and the body (denoted as $\mathsf{body}(\varphi)$) of the clause, respectively. A clause is *definite* if all the literals in its body are positive (namely atoms), and $\mathbb{P}$ is *definite* if all its clauses are definite. We write $\mathbb{P}' \subseteq \mathbb{P}$ to mean that $\mathbb{P}'$ is a sub-program of $\mathbb{P}$, and $\mathcal{P}(\mathbb{P})$ for the set of all sub-programs of $\mathbb{P}$.

An *interpretation* $\mathcal{I}$ is a subset of At, and it is a *model* of $\mathbb{P}$ if for all $\varphi \in \mathbb{P}$, $\mathsf{body}(\varphi) \subseteq \mathcal{I}$ implies $\mathsf{head}(\varphi) \in \mathcal{I}$. A literal L is true in $\mathcal{I}$, denoted as $\mathcal{I} \vDash L$, if either $L = B$ and $B \in \mathcal{I}$, or $L = \neg B$ and $B \notin \mathcal{I}$, for some $B \in At$. Suppose X is a set of literals, we define $\mathcal{I} \vDash X$

if $\mathcal{I} \vDash L$ for all $L \in X$. There is an inclusion ordering on models, and every definite logic program $\mathbb{P}$ has a least model (referred to as its least Herbrand semantics), denoted as $\mathcal{H}(\mathbb{P})$. Least models of definite logic programs can also be characterised as the least fixed points of *immediate consequence operators*. The immediate consequence operator $\mathbf{T}_{\mathbb{P}}$ associated to a logic program $\mathbb{P}$ with atom set At is a function $\mathcal{P}(At) \to \mathcal{P}(At)$ such that for any interpretation $\mathcal{I}$, $\mathbf{T}_{\mathbb{P}}(\mathcal{I}) \coloneqq \{A \mid \exists \psi \in \mathbb{P}$ such that $\mathsf{head}(\psi) = A$ and $\mathcal{I} \vDash \mathsf{body}(\psi)\}$.

Note that arbitrary, non-definite logic programs may not have a least model. There are various alternative denotational semantics, including stratified semantics [3, 30], stable semantics [18], well-founded semantics [38]. We recall the stratified semantics from [3]. Given a LP $\mathbb{P}$, the *definition* of an atom A is the sub-program $def(A) \coloneqq \{\varphi \in \mathbb{P} \mid \mathsf{head}(\varphi) = A\}$. $\mathbb{P}$ is *stratified* if there exists a partition $At_1, \ldots, At_k$ (called a stratification) of At such that,

- If there exists $\mathbb{P}_i$-clause φ such that $\neg B \in \mathsf{body}(\varphi)$, then $def(B)$ only contains atoms in $\bigcup_{j<i} At_j$.
- If there exists $\mathbb{P}_i$-clause φ such that $B \in \mathsf{body}(\varphi)$, then $def(B)$ only contains atoms in $\bigcup_{j\leq i} At_j$.

where $\mathbb{P}_1, \ldots, \mathbb{P}_k$ is the partition of $\mathbb{P}$ induced by the stratification: each $\mathbb{P}_i$ is $\{\varphi \in \mathbb{P} \mid \mathsf{head}(\varphi) \in At_i\}$. Then the *stratified model* $\mathcal{S}(\mathbb{P})$ of $\mathbb{P}$ is defined as $\mathcal{S}(\mathbb{P}) = \cup_{i=1}^{k} M_i$, where $M_i \coloneqq \mathcal{H}(\mathbb{Q}_i)$, and each $\mathbb{Q}_i$ is obtained by (1) deleting all the $\mathbb{P}_i$ clauses whose bodies contain some literals false in $\cup_{j<i} M_j$, and (2) in the remaining clauses, delete all the literals not in At_i and the negated literals. Importantly, the stratified semantics is independent of the choice of a specific stratification.

Probabilistic logic programming. A *probabilistic logic program* (PLP) $\mathbb{P}$ is a set of probabilistic clauses ψ of the form $p :: \varphi$, where $p \in (0, 1]$ is a real number, and φ is a clause. p is called the (probabilistic) label of clause ψ, denoted as $\mathsf{Lab}(\psi)$. By forgetting the labels in $\mathbb{P}$, we obtain the pure logic program $|\mathbb{P}|$ of $\mathbb{P}$.

We will use the "bra-ket" (or Dirac) notation for probability distributions, e.g. $0.7|a\rangle + 0.3|b\rangle$ is the distribution on $\{a, b\}$ assigning probability 0.7 to a and 0.3 to b. We recall the distribution semantics. A PLP $\mathbb{P}$ determines a probability distribution $\mu_{\mathbb{P}}$ over the sub-programs of $|\mathbb{P}|$: for any $\mathbb{L} \subseteq |\mathbb{P}|$, $\mu_{\mathbb{P}}(\mathbb{L}) \coloneqq \left(\prod_{\varphi\in\mathbb{L}} \mathsf{Lab}(\varphi)\right) \cdot \left(\prod_{\varphi\in|\mathbb{P}|\setminus\mathbb{L}} (1 - \mathsf{Lab}(\varphi))\right)$. If $\mathbb{P}$ is acyclic, $\mu_{\mathbb{P}}$ allows to compute the probability that a given goal is proved. The *success probability* $\pi_{\mathbb{P}}(\bar{L})$ (or simply probability) of a goal $\bar{L} = L_1 \wedge \cdots \wedge L_m$ is $\sum \{\mu_{\mathbb{P}}(\mathbb{L}) \mid \mathbb{L} \subseteq |\mathbb{P}|, \mathcal{S}(\mathbb{L}) \vDash L_1, \ldots, L_m\}$. The distribution $\delta_{\mathbb{P}}(\bar{A})$ of a set of atoms $\bar{A} = \{A_1, \ldots, A_m\}$ over its interpretations is thus defined as $\sum_{L_i\in\{A_i,\neg A_i\}} \pi_{\mathbb{P}}(L_1 \wedge \cdots \wedge L_m)|L_1, \ldots, L_m\rangle$.

▶ **Example 1.** The PLP program $\mathbb{P}_{\text{wet}}$ below describes how the season affects the probability of raining and a sprinkler to leak, which cause the grass to be wet and the road to be slippery.

0.25 ::	`Winter`	← .	(ψ_1)	0.9 ::	`WetGrass`	← `Sprinkler`.	(ψ_5)
0.2 ::	`Sprinkler`	← `Winter`.	(ψ_2)	0.8 ::	`WetGrass`	← `Rain`.	(ψ_6)
0.6 ::	`Rain`	← `Winter`.	(ψ_3)	0.7 ::	`SlipperyRoad`	← `Rain`.	(ψ_7)
0.1 ::	`Rain`	← ¬`Winter`.	(ψ_4)	0.1 ::	`SlipperyRoad`	← ¬`Rain`.	(ψ_8)

If we forget about all the probabilistic labels, then we get a logic program $|\mathbb{P}_{\text{wet}}|$. For instance, we can calculate the success probability $\pi_{\mathbb{P}_{\text{wet}}}(\texttt{Winter} \wedge \texttt{WetGrass}) = 0.1434$.

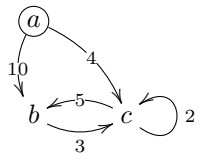
Weighted Logic Programming. A weighted logic program (WLP) $\mathbb{P}$ based on an ω-complete commutative semiring $\mathcal{K} = \langle K, +, \cdot, \mathbf{0}, \mathbf{1}\rangle$ is a finite set of weighted clauses φ of the form $w :: A \leftarrow B_1, \cdots, B_k.$, where $w \in K$ is the weight label of the clause (denoted as $lab(\varphi)$).

We also assume that $\mathcal{K}$ is a complete lattice under the ordering "$x \preceq y$ if $\exists u \in \mathcal{K}$ such that $x + u = y$". Given atoms $At = \{A_1, \ldots, A_n\}$, for each A_i, let $\mathcal{K}_{A_i}$ be a copy of $\mathcal{K}$ standing for the values of atom A_i, and we write $\mathcal{K}_{A_i}$ as $\{w_{A_i} \mid w \in \mathcal{K}\}$. A *weight state* is a tuple $u \in \mathcal{K}_{A_1} \times \cdots \times \mathcal{K}_{A_n}$, and its i-th component is referred to as $u(A_i)$.

The semantics of WLP assigns weights to atoms that are defined as the least fixed point of a weighted variant $\mathbf{T}^w$ of the immediate consequence operator for classical logic programs [15]. Given a WLP program $\mathbb{P}$ with atom set $\{A_1, \ldots, A_n\}$, $\mathbf{T}^w_{\mathbb{P}}$ is a function $\mathcal{K}_{A_1} \times \cdots \times \mathcal{K}_{A_n} \to \mathcal{K}_{A_1} \times \cdots \times \mathcal{K}_{A_k}$ such that for each weight state u, the i-th component of $\mathbf{T}^w_{\mathbb{P}}(u)$ is $\sum\{lab(\varphi) \cdot u(B_1) \cdot \cdots \cdot u(B_k) \mid \varphi \in \mathbb{P}, \mathsf{head}(\varphi) = A_i, \mathsf{body}(\varphi) = \{B_1, \ldots, B_k\}\}$. The *weight* $weight_{\mathbb{P}}(A_i)$ of A_i is then the i-th component of the least fixed point of $\mathbf{T}^w_{\mathbb{P}}$, which exists because $\mathbf{T}^w_{\mathbb{P}}$ is a monotonic function on an ω-complete lattice $\mathcal{K}_{A_1} \times \cdots \times \mathcal{K}_{A_n}$.

▶ **Example 2.** The semiring $\mathcal{K}^{sp} = \langle \mathbb{N} \cup \{+\infty\}, \min, +, +\infty, 0\rangle$ is a fragment of the "min-plus" tropical semiring, used for shortest path problems. Consider the WLP program $\mathbb{P}_{sp}$ consisting of all *the grounding of* the clauses below left in (1), which describes the reachability condition of the weighted directed graph D below right in (1). Then the answer to the shortest path from the initial state to a state `x` can be calculated by the weight of `reachable(x)` in $\mathbb{P}_{sp}$. For instance, $weight_{\mathbb{P}_{sp}}(\texttt{reachable(b)}) = 9$, which is the least path weight from `a` to `b` in D.

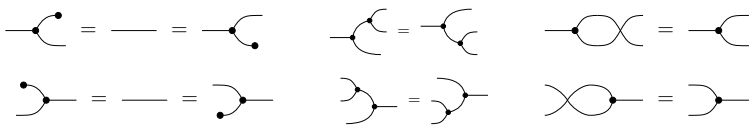
```
0 :: initial(a) ← .   | 10 :: edge(a,b) ← .
4 :: edge(a,c)  ← .   |  3 :: edge(b,c) ← .
5 :: edge(c,b)  ← .   |  2 :: edge(c,c) ← .
0 :: reachable(x) ← initial(x).
0 :: reachable(x) ← reachable(y), edge(y,x).
```



(1)

Bayesian networks. We will explore the relationship between PLP and Bayesian networks (BN), which we briefly recall. A Bayesian network $\mathbb{B}$ on a set $A_1, A_2, \ldots A_k$ of variables (finite sets) is a pair $(G, \Pr)$. Here G is a directed acyclic graph (DAG) (V_G, E_G), where $V_G = A_1, A_2, \ldots A_k$. The second component Pr is a family $\{\Pr(A \mid pa(A))\}_{A \in V_G}$ of conditional probability distributions, where $pa(A)$ is the set of predecessors of A according to E_G, and the distribution $\Pr(A \mid pa(A))$ assigns a probability to each element in the set A. We say that BN is *boolean-valued* when each variable A is a two-element set, which we write $\{A, \neg A\}$ with slight abuse of notation (meaning "A is true", "A is false").

CDMU and CD categories. We will formulate both the syntax and the semantics of logic program in terms of categories, equipped with the structure of a *CDMU category*. A CDMU category (for **c**opy, **d**iscard, **m**ultiplication, **u**nit) is a symmetric monoidal category $\langle \mathbf{C}, \otimes, I\rangle$ where each object C has a copier $\multimap_C : C \to C \otimes C$, a discarder $\multimap_C : C \to I$, a multiplication $\multimap_C : C \otimes C \to C$, and a unit $\multimap_C : I \to C$ – which we will often write omitting C, when clear from the context. These operations are required to satisfy the following set of equations:



(2)

In words, there are a commutative comonoid $(\multimap, \multimap)$ and a commutative monoid $(\multimap, \multimap)$. Moreover, the four operations are required to be compatible with the tensor structure on $\mathbf{C}$, in the expected way. A *CDMU functor* between two CDMU categories is a strong symmetric monoidal functor that also preserves the commutative comonoid and monoid structures. We will often use the construction of the *free* CDMU category $\mathsf{freeCDMU}(X, \Sigma)$ over a generating

set X of objects and a generating set Σ of morphisms: the objects are finite lists $[x_1, \ldots, x_n]$ over X (including the empty list $[\,]$), and the morphisms are string diagrams [37] freely obtained by composing morphisms in Σ together with $\multimap, \bullet\!\!-, \succ\!\!-, \bullet\!\!-$, modulo the equations in (2).

When describing Bayesian networks, we will also need to refer to *CD categories*. These are defined in the same way as CDMU categories, but without the multiplications $\succ\!\!-_C : C \otimes C \to C$ and units $\bullet\!\!-_C : I \to C$. We write $\mathsf{freeCD}(X, \Sigma)$ for the free CD category over objects X and morphisms Σ. See [22] and Appendix A for background on how CD categories relate to BNs.

3 Classical Logic Programming

In this section we introduce a *functorial semantics* of classical logic programs, which culminates in Proposition 5 below. Our starting point is a conceptual distinction between the syntax and the semantics of a logic program. In Subsection 3.1 we introduce the syntax category of string diagrams for a given logic program $\mathbb{P}$, which will also be used in Sections 4 and 5 for the probabilistic and the weighted case. Next, in Subsection 3.2 we provide a semantics category, and identifies logic programs with certain *models* – structure-preserving functors from the syntax category to the semantics category. Finally, in Subsection 3.3 we describe some well-known semantics (immediate consequence operator, least Herbrand semantics, stratified semantics) of logic programs as images of specific string diagrams in the syntax, under the functorial interpretation. This perspective will provide an original, diagrammatic representation for such semantic constructs.

3.1 Syntax Category

Every logic program $\mathbb{P}$ has an underlying definite logic program $[\![\mathbb{P}]\!]$ describing the purely inferential structure of $\mathbb{P}$ – where, intuitively, we disregard information about probabilities, weights, and negated clauses. $[\![\mathbb{P}]\!]$ is what is commonly used in the definition of the *dependency graph* of $\mathbb{P}$ [3], which describes the dependency relation between atoms in $\mathbb{P}$, and plays a key role in the definition of stratified logic programs [3]. To incorporate this step in our approach, first we define a "forgetful" function τ^{gen} (resp. τ^{pr}, τ^{wt}) from arbitrary (classical, probabilistic, or weighted) clauses to definite clauses as follows:

$$\begin{array}{lcrlcl}
\tau^{gen} & : & A & \leftarrow B_1, ..., B_k, \neg C_1, ..., \neg C_\ell. & \mapsto & A \leftarrow B_1, ..., B_k, C_1, ..., C_\ell. \\
\tau^{pr} & : & p :: A & \leftarrow B_1, ..., B_k, \neg C_1, ..., \neg C_\ell. & \mapsto & A \leftarrow B_1, ..., B_k, C_1, ..., C_\ell. \\
\tau^{wt} & : & w :: A & \leftarrow B_1, ..., B_k. & \mapsto & A \leftarrow B_1, ..., B_k.
\end{array}$$

In words, τ forgets the quantitative labels and the negation. The definite logic program $[\![\mathbb{P}]\!]$ is defined as the image of $\mathbb{P}$ under the appropriate τ, namely $[\![\mathbb{P}]\!] \coloneqq \{\tau(\varphi) \mid \varphi \in \mathbb{P}\}$. We say $\mathbb{P}$ is based on $[\![\mathbb{P}]\!]$, or $[\![\mathbb{P}]\!]$ is the underlying program of $\mathbb{P}$. Note that, if we read $A \leftarrow B_1, \ldots, B_k.$ as "nodes $B_1, \ldots, B_k$ are parents of A", $[\![\mathbb{P}]\!]$ defines exactly (the components of) the dependency graph of $\mathbb{P}$ – *cf.* [3]. Observe that there may exist distinct clauses in $\mathbb{P}$ that have the same image under τ, so the size of $[\![\mathbb{P}]\!]$ is less or equal to that of $\mathbb{P}$.

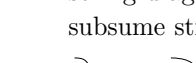
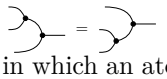
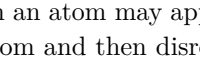
▶ **Example 3.** Recall $\mathbb{P}_{\text{wet}}$ from Example 1. The underlying definite logic program $[\![\mathbb{P}_{\text{wet}}]\!]$ is

`Winter`	$\leftarrow$.	(φ_1)	`WetGrass`	$\leftarrow$ `Sprinkler`.	(φ_4)
`Sprinkler`	$\leftarrow$ `Winter`.	(φ_2)	`WetGrass`	$\leftarrow$ `Rain`.	(φ_5)
`Rain`	$\leftarrow$ `Winter`.	(φ_3)	`SlipperyRoad`	$\leftarrow$ `Rain`.	(φ_6)

For instance, $\tau^{pr}(\psi_3) = \tau^{pr}(\psi_4) = (\texttt{Rain} \leftarrow \texttt{Winter}.) = \varphi_3$.

Given a definite logic program $\mathbb{L}$ on At, we construct a CDMU category $\mathsf{SynLP}_{\mathbb{L}}$ which encodes the inferential structure represented by $\mathbb{L}$. We define $\mathsf{SynLP}_{\mathbb{L}}$ as the free CDMU category $\mathsf{freeCDMU}(At, \Sigma_{\mathbb{L}})$ (*cf.* *Section 2*), where the set $\Sigma_{\mathbb{L}}$ of generating morphisms consists of one string diagram for each clause in $\mathbb{L}$:

$$\Sigma_{\mathbb{L}} \coloneqq \left\{ \begin{array}{c} B_1 \\ \vdots \\ B_m \end{array} \boxed{\varphi} \; A \;\middle|\; \varphi \equiv A \leftarrow B_1, \ldots, B_m. \text{ is a clause in } \mathbb{L} \right\} \tag{3}$$

With some abuse of notation, we use φ to refer to both a clause and its corresponding string diagram in $\Sigma_{\mathbb{L}}$. We choose to work with CDMU categories because their equations subsume structure that is always present in logic programs. For example,  and  reflect the intuition that there is no ordering on the (possibly multiple) clauses in which an atom may appear;  says that generating two occurrences of the same atom and then disregarding one is the same as just working with a single occurrence.

$\mathsf{SynLP}_{\mathbb{L}}$ will act as the syntax category for all $\mathbb{P}$ such that $[\mathbb{P}] = \mathbb{L}$. We will identify logic programs based on $\mathbb{L}$ with functors from $\mathsf{SynLP}_{\mathbb{L}}$ to some appropriate "semantics categories", which will vary depending on whether the program is classical, probabilistic, or weighted.

3.2 Functorial Semantics of LP

In this subsection we introduce a categorical semantics for classical logic programming, and characterise logic programs as functors to this semantics.

For the classical case, the semantics domain will be the category $\mathbf{Set}(\mathbf{2})$ defined as follows. Objects of $\mathbf{Set}(\mathbf{2})$ are finite products $\mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_n}$, where each $\mathbf{2}_{A_i}$ is a two-element Boolean algebra, which we write $\{A_i, \neg A_i\}$, with $A_i > \neg A_i$, to emphasise that the two elements will be treated as an atom and its negation. In particular, the singleton set $\mathbf{1} = \{*\}$ is the 0-ary product. Morphisms in $\mathbf{Set}(\mathbf{2})$ are simply functions between the underlying sets.

We write $\vee$, $\wedge$, $(\cdot)^-$ for the standard Boolean algebra operations. Note every $\mathbf{Set}(\mathbf{2})$-object is itself a finite Boolean algebra, with the operations defined pointwise.

Given a classical logic program $\mathbb{P}$ with underlying definite program $\mathbb{L} \coloneqq [\mathbb{P}]$, we may associate $\mathbb{P}$ with a functor $[\![-]\!]_{\mathbb{P}} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$ defined as follows. On objects, $[\![-]\!]_{\mathbb{P}}$ maps $A \in At$ to $\mathbf{2}_A = \{A, \neg A\}$. On morphisms, the CDMU structure is interpreted as

$$\begin{array}{llllll} [\![\multimap_A]\!]_{\mathbb{P}} : & \mathbf{2}_A \to \mathbf{2}_A \times \mathbf{2}_A & [\![\multimap_A]\!]_{\mathbb{P}} : & \mathbf{2}_A \to \mathbf{1} & [\![\multimap_A]\!]_{\mathbb{P}} : & \mathbf{2}_A \times \mathbf{2}_A \to \mathbf{2}_A \\ & x \mapsto (x, x) & & x \mapsto * & & (x, y) \mapsto x \vee y \end{array} \qquad [\![\multimap_A]\!]_{\mathbb{P}} : \mathbf{1} \to \mathbf{2}_A \quad * \mapsto \neg A.$$

For each $\mathbb{L}$-clause $\varphi \equiv A \leftarrow B_1, \ldots, B_m.$, $[\![\varphi]\!]_{\mathbb{P}}$ maps $u \in \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$ to A if u as an interpretation (*cf.* Sec. 2) satisfies $u \vDash \mathsf{body}(\psi)$ for some $\psi \in \mathbb{P}$ with $\tau^{gen}(\psi) = \varphi$, and to $\neg A$ otherwise.

Note that, in order for $[\![-]\!]_{\mathbb{P}}$ to be well-defined, the semantic domain $\mathbf{Set}(\mathbf{2})$ should also satisfy the CDMU equations – see Lemma 27 in Appendix B for a proof.

▶ **Example 4.** The clause $\psi \equiv A \leftarrow B_1, \neg B_2.$ is associated with the definite clause $\varphi \equiv A \leftarrow B_1, \neg B_2.$, and it gets interpreted as a function $[\![\varphi]\!] : \mathbf{2}_{B_1} \times \mathbf{2}_{B_2} \to \mathbf{2}_A$ mapping $(B_1, \neg B_2)$ to A and (B_1, B_2) to $\neg A$. Incidentally, note this is *not* a boolean function, as for instance $(B_1, \neg B_2) \vee (B_1, B_2) = (B_1, B_2)$ but $[\![\varphi]\!]\,(B_1, \neg B_2) \vee [\![\varphi]\!]\,(B_1, B_2) \neq [\![\varphi]\!]\,(B_1, B_2)$.

The next proposition formally states the correspondence characterising the functorial semantics of LP. In there, we call *generator-preserving* a functor $\mathbf{C} \to \mathbf{D}$ that maps generating objects of $\mathbf{C}$ to generating objects of $\mathbf{D}$ (assuming objects of the two categories are freely obtained from a set of generators). In our case, $\mathbf{C} = \mathsf{SynLP}_{\mathbb{L}}$, $\mathbf{D} = \mathbf{Set}(\mathbf{2})$ and the requirement ensures that a functor maps each atom $A \in At$, seen as object of $\mathsf{SynLP}_{\mathbb{L}}$, to a two-element Boolean algebra $\mathbf{2}$, which we write $\mathbf{2}_A = \{A, \neg A\}$ to emphasise this correspondence.

▶ **Proposition 5.** *There is a* 1-1 *correspondence between logic programs based on* $\mathbb{L}$ *and generator-preserving CDMU functors of type* $\mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$.

Echoing the terminology of Lawvere's functorial semantics [29], we will refer to the functors as in Proposition 5 simply as *models* in $\mathbf{Set}(\mathbf{2})$ of the syntactic theory $\mathsf{SynLP}_{\mathbb{L}}$.

3.3 A Gallery of diagrammatic representations of semantic constructs

A pleasant outcome of the functorial semantics is the possibility of capturing some well-known semantic approaches to logic programs using the diagrammatic language. More precisely, we will study string diagrams in the syntax category whose images under the model are the semantics we are interested in. We shall discuss three of them: the immediate consequence operator, the least Herbrand semantics, and the stratified semantics.

Immediate consequence operator

We begin with the immediate consequence operator $\mathbf{T}$ (*cf. Section 2*), a basic yet fundamental concept beneath many denotational semantics of logic programs [17, 3, 18, 38]. Let us fix a logic program $\mathbb{P}$ with atoms $At = \{A_1, \ldots, A_n\}$ and $\mathbb{L} := [\mathbb{P}]$. Intuitively, given an interpretation $\mathcal{I}$, $\mathbf{T}_{\mathbb{P}}(\mathcal{I})$ is the set of atoms derivable from the set $\mathcal{I}$ of "assumptions" using each $\mathbb{P}$-clause exactly once, in parallel. Note that $\mathbf{T}_{\mathbb{P}}$ is not extensive, in the sense that $A \in \mathcal{I}$ does not imply $A \in \mathbf{T}_{\mathbb{P}}(\mathcal{I})$. Also, note there is a canonical isomorphism between $\mathcal{P}(At)$ and $\mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_n}$, which we will exploit to formulate the operator as a morphism in $\mathbf{Set}(\mathbf{2})$.

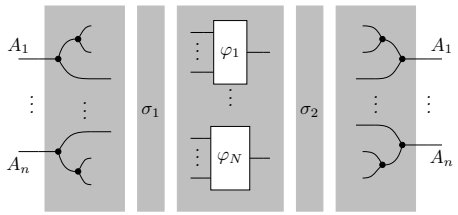
▶ **Example 6.** We consider an example from [18]. The atom set At_{pq} consists of $p(1,1)$, $p(1,2)$, $p(2,1)$, $p(2,2)$, $q(1)$, $q(2)$, and the program $\mathbb{P}_{pq}$ contains the following six clauses:

$$\begin{array}{lll} \psi_1 \equiv p(1,2) \leftarrow . & \psi_3 \equiv q(1) \leftarrow p(1,1), \neg q(1). & \psi_5 \equiv q(2) \leftarrow p(2,1), \neg q(1). \\ \psi_2 \equiv p(2,1) \leftarrow . & \psi_4 \equiv q(1) \leftarrow p(1,2), \neg q(2). & \psi_6 \equiv q(2) \leftarrow p(2,2), \neg q(2). \end{array}$$

Then $\mathbf{T}_{\mathbb{P}_{pq}}$ is a function $\mathcal{P}(At_{pq}) \to \mathcal{P}(At_{pq})$ which, for instance, maps both $\varnothing$ and $\{p(1,2), q(2)\}$ to $\{p(1,2), p(2,1)\}$, and maps $\{p(1,2)\}$ to $\{p(1,2), p(2,1), q(1)\}$.

We can express $\mathbf{T}_{\mathbb{P}}$ via our diagrammatic language by putting side-by-side all the generating morphisms in $\Sigma_{\mathbb{L}}$, one for each clause.

▶ **Proposition 7.** *The following string diagram* $t_{[\mathbb{P}]}$ *(in* $\mathsf{SynLP}_{[\mathbb{P}]}$*) satisfies* $[\![t_{[\mathbb{P}]}]\!]_{\mathbb{P}} = \mathbf{T}_{\mathbb{P}}$.



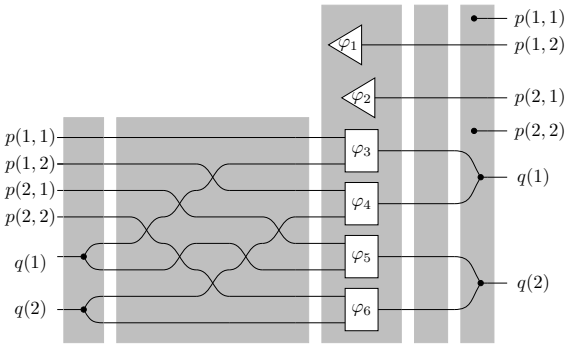
(4)

The left and the right ports both consist of all the atoms $A_1, \ldots, A_n$ *in At. For each* A_i*, suppose there are* k_i*-many clauses whose bodies include* A_i *and* ℓ_i*-many clauses whose heads are* A_i*. We make* k_i *copies (via* ⊸◂ *in the left-most box) and* ℓ_i *cocopies (via* ▸⊸ *in the right-most box) of* A_i*. The middle box contains the parallel composition of all* $\mathbb{L}$*-clauses* $\varphi_1, \ldots, \varphi_N$*. In the two* σ_i*-boxes, we have suitably many swapping morphisms* $\times$ *to match each copy/cocopy of* A_i *with an input/output wire* A_i *in the middle box.*

▶ **Example 8.** The underlying definite program of $\mathbb{P}_{pq}$ in Example 6 consists of six definite clauses:

$$\varphi_1 \equiv p(1,2) \leftarrow . \quad \varphi_3 \equiv q(1) \leftarrow p(1,1), q(1). \quad \varphi_5 \equiv q(2) \leftarrow p(2,1), q(1).$$
$$\varphi_2 \equiv p(2,1) \leftarrow . \quad \varphi_4 \equiv q(1) \leftarrow p(1,2), q(2). \quad \varphi_6 \equiv q(2) \leftarrow p(2,2), q(2).$$

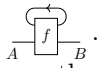
These clauses also constitute the set of generating morphisms $\Sigma_{\mathbb{P}_{pq}}$ of $\mathsf{SynLP}_{\mathbb{P}_{pq}}$. Then the corresponding string diagram $t_{[\mathbb{P}_{pq}]}$ for $\mathbf{T}_{\mathbb{P}_{pq}}$ is:



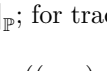
Interlude: traced extension

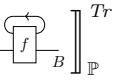
The last two semantic constructs we will consider are *fixed point-style* semantics: to provide the same kind of analysis as for the immediate consequence operator, we need to mildly extend our string diagrammatic language to include 'feedback wires' (called traces), and also extend the semantic category accordingly. The full picture of this extension is in (5) below.

▶ Remark 9. We purposefully choose not to include traces in the "basic" syntax category (Sec. 3.1), as we consider fixed points a feature specific to certain semantic constructs, rather than a primitive construction of logic program syntax. Also, note that traces are not required for our next developments in case one restricts attention to *acyclic* programs. This is mostly evident in the probabilistic case, where the distribution semantics is only defined for acyclic programs, and indeed the modelisation only uses the basic syntax, without traces.

We briefly recall traced categories, referring to [2] for full details. Recall that a symmetric monoidal category $\langle \mathbf{C}, \otimes, I \rangle$ is *traced* if it is equipped with a natural family of functions $Tr^X_{A,B}(f) \colon \mathbf{C}(X \otimes A, X \otimes B) \to \mathbf{C}(A, B)$ satisfying certain compatibility conditions [2]. In string diagrams, $Tr^X_{A,B}(f)$ is depicted as 

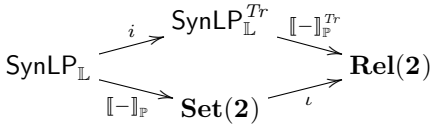
. We will use the free construction $\mathsf{freeTrCDMU}(X, \Sigma)$ of a traced CDMU category: the objects are finite lists over X, the morphisms are obtained as string diagrams in $\mathsf{freeCDMU}(X, \Sigma)$ plus the possibility of adding feedback loops, modulo the axioms for traced categories (*cf.* [1] for a detailed definition). The syntax category for $\mathbb{P}$ is defined as $\mathsf{freeTrCDMU}(At, \Sigma_{\mathbb{L}})$, where $\mathbb{L} = [\mathbb{P}]$. As for the semantics categories, we move to the category $\mathbf{Rel}(\mathbf{2})$ whose objects are the same as $\mathbf{Set}(\mathbf{2})$ and morphisms $A \to B$ are *relations* $R \subseteq A \times B$. The CDMU structure on $\mathbf{Rel}(\mathbf{2})$ consists of the graphs of that on $\mathbf{Set}(\mathbf{2})$. In fact, $\mathbf{Rel}(\mathbf{2})$ is a compact closed category – where we let the cartesian product act as the monoidal product – and thus equipped with a canonical traced structure (inherited from the category of relations).

Each logic program $\mathbb{P}$ uniquely determines a traced CDMU functor $[\![-]\!]^{Tr}_{\mathbb{P}}$, inductively defined on the generating morphisms as follows: for every trace-free string diagram f, $[\![f]\!]^{Tr}_{\mathbb{P}}$ is the relation describing the graph of the function $[\![f]\!]_{\mathbb{P}}$; for traced diagrams, given 

, 

$\Big]\!\Big]^{Tr}_{\mathbb{P}}$ is $\{(a,b) \in [\![A]\!]^{Tr}_{\mathbb{P}} \times [\![B]\!]^{Tr}_{\mathbb{P}} \mid \exists x \in [\![X]\!]^{Tr}_{\mathbb{P}} : ((x,a),(x,b)) \in [\![f]\!]^{Tr}_{\mathbb{P}}\}$.

The relationship between $[\![-]\!]_{\mathbb{P}}$ and $[\![-]\!]_{\mathbb{P}}^{Tr}$ can be summarised as the commutative square (5), where i identifies $\mathsf{SynLP}_{\mathbb{L}}$ as a subcategory of $\mathsf{SynLP}_{\mathbb{L}}^{Tr}$, and ι maps a function to its graph.

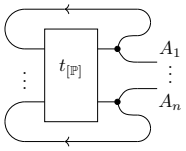


(5)

Herbrand semantics

The importance of the immediate consequence operator $\mathbf{T}_{\mathbb{P}}$ lies in that it performs the single "iteration step" in various denotational semantics of logic programs. For definite logic programs, the least models are precisely the least fixed points of the immediate consequence operator, which exist because the operators are monotonic on the complete lattice $\mathcal{P}(At)$. This suggests that one can express the *least Herbrand model* $\mathcal{H}$ (*cf.* Section 2) simply as immediate consequence operator plus "feedback wires".

▶ **Proposition 10.** *Suppose* $\mathbb{P}$ *is definite. The string diagram* $h_{[\mathbb{P}]}$ *in* $\mathsf{Syn}^{cb}_{[\mathbb{P}]}$ *defined in* (6) *expresses the least Herbrand model of* $\mathbb{P}$*, in the sense that* $\mathcal{H}(\mathbb{P}) = \min [\![h_{[\mathbb{P}]}]\!]_{\mathbb{P}}^{Tr}$*, where the box is* (4)*.*



(6)

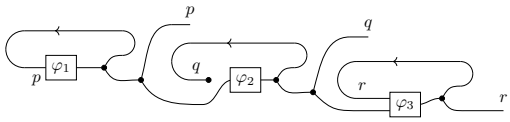
Observe that, even though the least Herbrand model is just one element of $[\![h_{[\mathbb{P}]}]\!]_{\mathbb{P}}^{Tr}$ in (6), as a whole $[\![h_{[\mathbb{P}]}]\!]_{\mathbb{P}}^{Tr}$ still expresses a well-known semantic concept, namely the set of *supported models* of the logic program $\mathbb{P}$. An interpretation $\mathcal{I}$ is a *model* of $\mathbb{P}$ if for each $\mathbb{P}$-clause ψ, $\mathcal{I} \vDash \mathsf{body}(\psi)$ implies $\mathsf{head}(\psi) \in \mathcal{I}$; it is *supported* if for each $A \in \mathcal{I}$, there exists some $\mathbb{P}$-clause ψ such that $\mathsf{head}(\psi) = A$ and $\mathcal{I} \vDash \mathsf{body}(\psi)$. Note that supported models are exactly the fixed points of $\mathbf{T}_{\mathbb{P}}$ [33]. It then follows immediately that the string diagram $h_{[\mathbb{P}]}$ (6) expresses the supported models of $\mathbb{P}$: $[\![h_{[\mathbb{P}]}]\!]_{\mathbb{P}}^{Tr} = \{\mathcal{I} \mid \mathcal{I} \text{ is a supported model of } \mathbb{P}\}$.

Stratified semantics

We move on to the case of stratified semantics of stratified logic programs (*cf.* Section 2). Suppose program $\mathbb{P}$ is stratified, where $At_1, \cdots, At_k$ is a stratification of At, and $\mathbb{P}_1, \ldots, \mathbb{P}_k$ is the associated partition of $\mathbb{P}$. The idea of stratification is to turn $\mathbb{P}_1, \ldots, \mathbb{P}_k$ into definite logic programs whose least models together form a model of the original program $\mathbb{P}$. As recalled in Section 2, this is achieved by observing that $\mathbb{P}_1$ is always definite by definition, and $\mathbb{P}_i$ for $i > 1$ can be turned into a definite program using the least models for the definite programs of $\{\mathbb{P}_j\}_{j<i}$. This idea suggest the construction of a string diagram representing stratified semantics in a layer-by-layer style: the output wires of all $\{h_{[\mathbb{P}_j]}\}_{j<i}$ (defined in Proposition 10) will serve as inputs of $h_{[\mathbb{P}_i]}$. For the sake of clarity, rather than detailing the fully general case of this construction, we believe it is best illustrated via an example.

▶ **Example 11.** Consider the program $\mathbb{O}$ consisting of clauses $\psi_1 \equiv p \leftarrow p$, $\psi_2 \equiv q \leftarrow \neg p$ and $\psi_3 \equiv r \leftarrow r, \neg q$. One stratification is $P_1 = \{p\}$, $P_2 = \{q\}$, $P_3 = \{r\}$, whose corresponding partition of $\mathbb{O}$ is $\mathbb{O}_1 = \{p \leftarrow p.\}$, $\mathbb{O}_2 = \{q \leftarrow \neg p.\}$, $\mathbb{O}_3 = \{r \leftarrow r, \neg q.\}$. Let φ_i be $\tau^{gen}(\psi_i)$, for $i = 1, 2, 3$.

The string diagram $s_{[\mathbb{O}]}$ below expresses the stratified model $\{q\}$ of $\mathbb{O}$, in the sense that the stratified model is the least element under the lexicographical order in $[\![s_{[\mathbb{O}]}]\!]_{\mathbb{O}}$: for any $(i_p, i_q, i_r) \in [\![s_{[\mathbb{O}]}]\!]_{\mathbb{O}}$, there are three possibilities, either $\neg p \leq i_p$, or $\neg p = i_p$ and $q \leq i_q$, or $\neg p = i_p$, $q = i_q$ and $\neg r \leq i_r$.



4 Probabilistic Logic Programming

In this section we turn to probabilistic logic programming (PLP). We first give the functorial semantics of PLP in Section 4.1. Next we discuss a pictorial representation of distribution semantics, in Section 4.2. A benefit of the diagrammatic representation of PLP syntax as string diagrams is that the correspondence with a closely related formalism, namely *Bayesian networks* (BNs), become more apparent: we explore this in Section 4.3, where we show the equivalence between boolean-valued BNs and acyclic PLP and illustrate their relationship with functors between the corresponding categories.

For PLP, the separation of syntax and semantics into different categories provides two main insights: first, we are able to define PLP programs as models of the same syntax category as LP programs, thus formalising the intuition that the difference between the two formalisms is only at the semantics level. Second, in order to formalise the correspondence between PLP programs and boolean-valued BNs, we only need to act on the syntactic categories for the two formalisms, thus showing that the two rely on the same semantic layer, and differ in how this is described by syntactic/combinatorial structures.

4.1 Functorial Semantics of PLP

Throughout this section we fix an acyclic definite logic program $\mathbb{L}$. To be precise, by "$\mathbb{L}$ being acyclic" we mean that there is no finite sequence of $\mathbb{L}$-clauses $\varphi_0, \ldots, \varphi_m$ such that $\mathsf{head}(\varphi_{i+1}) \in \mathsf{body}(\varphi_i)$ for all $i = 0, \ldots, m-1$, and $\mathsf{head}(\varphi_0) \in \mathsf{body}(\varphi_m)$. Our goal is to provide for PLP a functorial semantics characterisation analogous to the one of Proposition 5. As mentioned above, we will use the same syntax category as for LP, introduced in Section 3.1. What differs is the semantics category, which we now introduce. Intuitively, it amounts to switching from boolean-valued functions to their probabilistic counterpart.

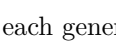
▶ **Definition 12.** *The category* $\mathbf{Stoch}(\mathbf{2})$ *is defined as having the same objects as* $\mathbf{Set}(\mathbf{2})$*, and morphisms the functions of the form* $f \colon \mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_k} \to \mathcal{D}(\mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m})$*, where* $\mathcal{D}(\mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m})$ *is the set of probability distributions on* $\mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$*. Given morphisms* $f \colon X \to \mathcal{D}(Y)$ *and* $g \colon Y \to \mathcal{D}(Z)$*, their composition* $g \circ f$ *assigns to each* $x \in X$ *a distribution* $\sum_{z \in Z} \left(\sum_{y \in Y} f(x)(y) \cdot g(y)(z) \right) |z\rangle$.

Equivalently, $\mathbf{Stoch}(\mathbf{2})$ is the full subcategory of $\mathbf{Stoch}$ (the category of stochastic matrices) whose objects are those of $\mathbf{Set}(\mathbf{2})$. Also, $\mathbf{Stoch}$ is the same as the Kleisli category for the distribution monad, so we may regard composition in $\mathbf{Stoch}(\mathbf{2})$ simply as Kleisli composition.

Given a PLP $\mathbb{P}$ with $[\mathbb{P}] = \mathbb{L}$, we define a functor $[\![-]\!]_{\mathbb{P}} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Stoch}(\mathbf{2})$ as follows. On objects, $[\![-]\!]_{\mathbb{P}}$ maps $A \in At$ to $\mathbf{2}_A$. On morphisms, for the CDMU structure we define

$$[\![\multimap_A]\!]_{\mathbb{P}} : \mathbf{2}_A \to \mathbf{2}_A \times \mathbf{2}_A \quad x \mapsto 1|(x,x)\rangle \qquad [\![\multimap\!\bullet_A]\!]_{\mathbb{P}} : \mathbf{2}_A \to \mathbf{1} \quad x \mapsto 1|*\rangle$$

$$[\![\bullet\!\multimap_A]\!]_{\mathbb{P}} : \mathbf{2}_A \times \mathbf{2}_A \to \mathbf{2}_A \quad (x,y) \mapsto 1|x \vee y\rangle \qquad [\![\bullet\!\multimap_A]\!]_{\mathbb{P}} : \mathbf{1} \to \mathbf{2}_A \quad * \mapsto 1|\neg A\rangle$$

For each generating morphism  in $\mathsf{SynLP}_{\mathbb{L}}$, $[\![\varphi]\!]_{\mathbb{P}}$ maps a state $u \in \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$ to $p|A\rangle + (1-p)|\neg A\rangle$ if there exists a $\mathbb{P}$-clause ψ such that $u \vDash \mathsf{body}(\psi)$, $\tau^{pr}(\psi) = \varphi$ and $lab(\psi) = p$, and to $1|\neg A\rangle$ otherwise. As in the classical case, for $[\![-]\!]_{\mathbb{P}}$ to be well-defined, we need the semantics category $\mathbf{Stoch}(\mathbf{2})$ to satisfy the CDMU equations as shown in Appendix B, Lemma 28. We also refer to Appendix B for the proof of our characterisation result, which essentially follows the same steps as the proof of Proposition 5.

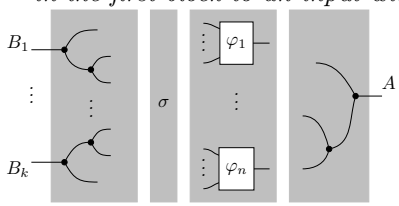
▶ **Proposition 13.** *There is a 1-1 correspondence between PLP programs based on $\mathbb{L}$ and generator-preserving CDMU functors $\mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Stoch}(\mathbf{2})$.*

4.2 Distribution Semantics in String Diagrams

As for LP, we now study diagrammatic representations of PLP semantics. As a preliminary, we record the notion of 'A-component'. Intuitively the A-component collects all the clauses with heads A, and "bundle" them into a string diagram with a single output A.

▶ **Definition 14.** *The A-component of $\mathbb{P}$ (based on $\mathbb{L}$) is a string diagram $comp_A \colon [B_1, \ldots, B_k] \to [A]$ in $\mathsf{SynLP}_{\mathbb{L}}$ constructed as below, where:*

- **(I)** *in the first block, for each atom B_i appearing in the body of some φ with $\mathsf{head}(\varphi) = A$, there are k_i-many copies of B_i (via ⊸), where k_i is the number of $\mathbb{L}$-clauses φ satisfying $\mathsf{head}(\varphi) = A$ and $B \in \mathsf{body}(\varphi)$.*
- **(II)** *In the third block, we have parallel string diagrams for each $\mathbb{L}$-clause $\varphi_1, \ldots, \varphi_n$ with head A.*
- **(III)** *The fourth block hosts n-many cocopies of A (via ⊸), where n is the number of $\mathbb{L}$-clauses with head A.*
- **(IV)** *The σ-block contains suitably many swapping morphisms $\mathbb{X}$ to match each copy of B_i in the first block to an input wire B_i in the third block.*



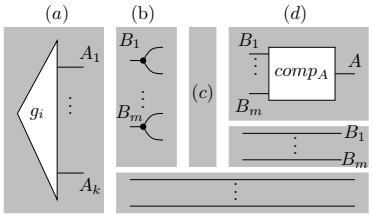
Note that, if $n = 0$ (namely there is no $\mathbb{L}$-clause with head A), then $comp_A$ simplifies to ⊸.

Now the goal is to diagrammatically express the probability distribution $\delta(\bar{A})$ of a set of atoms $\bar{A} \coloneqq \{A_1, \cdots, A_k\}$. The idea is to construct a string diagram $f_{\bar{A}} \colon [\,] \to [A_1, \ldots, A_k]$ which exhausts all possible derivations of $\bar{A}$ in $\mathbb{P}$. To do so, we will first define f_{At} and then obtain $f_{\bar{A}}$ by discarding all the atoms not appearing in $\bar{A}$. Note that the termination of the following construction relies on $\mathbb{P}$ being acyclic, and $comp_A$ is defined in Definition 14.

▶ **Construction 15.** Suppose $At = \{A_1, \ldots, A_n\}$. The string diagram $f_{At} \colon [\,] \to [A_1, \ldots, A_n]$ is defined via a step-by-step construction of morphisms g_i for $i \in \mathbb{N}$:

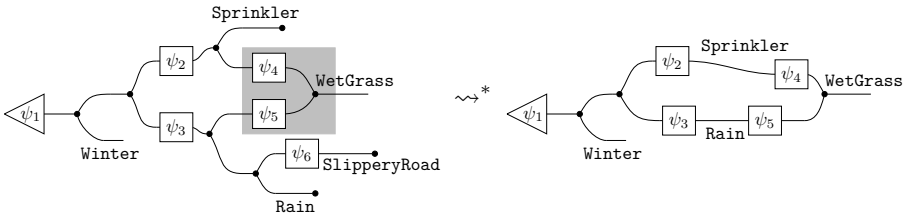
1. $g_0 = id_{[\,]} \colon [\,] \to [\,]$.
2. If g_i is of type $[\,] \to [A_1, \ldots, A_n]$, then let $f_{At} = g_i$.
3. Otherwise, g_i is of type $[\,] \to [A_1, \ldots, A_k]$, and we define g_{i+1} as the string diagram below. We pick any $A \in At \setminus \{A_1, \ldots, A_k\}$ such that all atoms in the domain of $comp_A$ already appear in $A_1, \ldots, A_k$, say $comp_A \colon [B_1, \ldots, B_m] \to [A]$ with $\{B_1, \ldots, B_m\} \subseteq \{A_1, \ldots, A_k\}$. Then in block (b) we make 2 copies for each $B_j \in \{B_1, \ldots, B_m\}$ from the

codomain of g_i in block (a), compose them with suitably many swapping morphisms $\times$ in block (c), and match one copy of each B_j to exactly one B_j from the domain of $comp_A$ in block (d).



We are now ready to construct the diagram $f_{\bar{A}}$. First, for each atom $B \in At \backslash \{A_1, \ldots, A_k\}$, discard B in f_{At} by post-composing $\multimap_B$. Next, simplify the diagram by the rewriting rules  $\rightsquigarrow$ —, $\rightsquigarrow$ —, and $\rightsquigarrow$ —. Note that while the first two rewriting rules are CDMU axioms, the third rewriting rule is not valid in $\mathsf{SynBN}_{\mathbb{L}}$. However, all of them are valid in the semantics category $\mathbf{Stoch}(\mathbf{2})$ [22], which justifies our procedure.

▶ **Example 16.** Recall the program $\mathbb{P}_{\text{wet}}$ from Example 1. The string diagram below left is $f_{At_{\text{wet}}}$ with atoms `Sprinkler`, `Rain` and `SlipperyRoad` discarded.



In particular, the subdiagram in the grey block is the `WetGrass`-component of type $[\texttt{Sprinkler}, \texttt{Rain}] \to [\texttt{WetGrass}]$. The string diagram $f_{\bar{A}}$ below right is the result of applying the three rewriting rules. One can verify that $[\![f_{\bar{A}}]\!]_{\mathbb{P}_{\text{wet}}}$ is the probability distribution $\delta(\texttt{Winter}, \texttt{WetGrass})$.

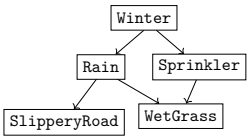
▶ **Proposition 17.** *$f_{\bar{A}}$ calculates the success probability:* $[\![f_{\bar{A}}]\!]_{\mathbb{P}} = \delta(\bar{A})$.

4.3 Correspondence of PLP and BNs via Functorial Semantics

We now turn attention to the relationship between PLP and Bayesian networks. Our goal is formulating the translation between the two formalisms at the functorial level, taking advantage on the one side of the functorial semantics of PLP just established, and on the other side of the functorial semantics of Bayesian networks, as described in [22].

It is known in the LP literature that every acyclic PLP can be transformed into an equivalent boolean-valued BN, and vice versa [31]. Intuitively, starting from a BN $\mathbb{B} = (G, \Pr)$ (*cf.* Section 2), one constructs a PLP program with a clause for each conditional probability in Pr. Conversely, given a PLP program $\mathbb{P}$, one constructs a DAG in which B is a parent of A precisely when there *exists* $\psi \in \mathbb{P}$ such that $A = \mathsf{head}(\psi)$ and B appears in $\mathsf{body}(\psi)$. Each conditional probability $\Pr(A = 1 \mid pa(A) = u)$ is calculated by "summing up" the probabilities of all $\mathbb{P}$ clauses ψ such that $\mathsf{head}(\psi) = A$ and $u \vDash \mathsf{body}(\psi)$ as independent random events.

▶ **Example 18.** Consider the PLP $\mathbb{P}_{\text{wet}}$ from Example 1. On the left below we show the DAG of the corresponding Bayesian network, and on the right the conditional probability associated with variable `WetGrass`.



		Pr(WetGrass)
¬Sprinkler	¬Rain	0
¬Sprinkler	Rain	0.8
Sprinkler	¬Rain	0.9
Sprinkler	Rain	0.98

(7)

Conversely, we may construct a PLP $\mathbb{P}'_{\text{wet}}$ corresponding to $\mathbb{B}_{\text{wet}}$, following the recipe given above. It differs with $\mathbb{P}_{\text{wet}}$ only on the clauses with heads `WetGrass`. Instead of ψ_5 and ψ_6 in Example 1, $\mathbb{P}'_{\text{wet}}$ has the following clauses which together specify the conditional probability in Figure (7):

$$0.8 :: \texttt{WetGrass} \leftarrow \neg\texttt{Sprinkler}, \texttt{Rain}. \qquad 0.9 :: \texttt{WetGrass} \leftarrow \texttt{Sprinkler}, \neg\texttt{Rain}.$$
$$0.98 :: \texttt{WetGrass} \leftarrow \texttt{Sprinkler}, \texttt{Rain}.$$

We now wish to study this two-way translation at the functorial level. To do so, we exploit the functorial semantics for BNs, as established in [22] (see also [16]), and reported in Appendix A. In a nutshell, this characterisation associates Bayesian networks based on a DAG $G = (V_G, \Sigma_G)$ with models of a freely generated syntax category $\mathsf{SynBN}_G := \mathsf{freeCD}(V_G, \Sigma_G)$, where intuitively the edges of G act as the generators of the diagrammatic syntax. For comparing it to PLP, we need a restriction of this characterisation result to *boolean-valued* Bayesian networks (*cf.* Section 2), which we report below.

▶ **Proposition 19** ([22]). *There is 1-1 correspondence between boolean-valued Bayesian networks based on a DAG G and generator-preserving CD functors $\mathsf{SynBN}_G \to \mathbf{Stoch}(\mathbf{2})$.*

We now describe the two-way translation between PLP and BNs, exploiting their view as models as provided by Propositions 13 and 19.

BN to PLP. Given a boolean-valued BN $\mathbb{B} = (G, \Pr)$ and its corresponding BN model $[\![-]\!]_{\mathbb{B}} : \mathsf{SynBN}_G \to \mathbf{Stoch}(\mathbf{2})$, we let the syntax category $\mathsf{SynLP}_{\mathbb{L}'}$ of the corresponding logic program be $\mathsf{freeCDMU}(V_G, \Sigma_G)$. Intuitively, this means that every node A in G yields exactly one clause $A \leftarrow pa(A)$ in $\mathbb{L}'$. Then we can define a PLP program $\mathbb{P}'$ via Proposition 13 as the CDMU functor $[\![-]\!]^b_{\mathbb{B}} : \mathsf{SynLP}_{\mathbb{L}'} \to \mathbf{Stoch}(\mathbf{2})$ obtained by canonically lifting the CD functor $[\![-]\!]_{\mathbb{B}} : \mathsf{SynBN}_G \to \mathbf{Stoch}(\mathbf{2})$ along the inclusion functor $\iota \colon \mathsf{SynBN}_G \to \mathsf{SynLP}_{\mathbb{L}'}$ which embeds the CD structure of $\mathsf{SynBN}_G = \mathsf{freeCD}(V_G, \Sigma_G)$ into the CDMU structure of $\mathsf{SynLP}_{\mathbb{L}'} = \mathsf{freeCDMU}(V_G, \Sigma_G)$. In concrete, $[\![-]\!]^b_{\mathbb{B}}$ maps $\succ\!\!-, \bullet\!\!-$ to the monoid structure in $\mathbf{Stoch}(\mathbf{2})$ and just behaves the same as $[\![-]\!]_{\mathbb{B}}$ on the rest. The construction of $[\![-]\!]^b_{\mathbb{B}}$ is summarised by the commutative square ② in (9) below. We can verify that this lifting $[\![-]\!]^b_{\mathbb{B}}$ indeed coincides with the model $[\![-]\!]_{\mathbb{P}'} : \mathsf{SynLP}_{\mathbb{L}'} \to \mathbf{Stoch}(\mathbf{2})$ associated with $\mathbb{P}'$, the PLP program that one could obtain from $\mathbb{B}$ in the traditional way (e.g. in [31]).

▶ **Proposition 20.** *Let $\mathbb{P}'$ be the PLP program encoding $\mathbb{B}$. Then $\Sigma_{[\mathbb{P}']} = \Sigma_{\mathbb{L}'}$ and $[\![-]\!]_{\mathbb{P}'} = [\![-]\!]^b_{\mathbb{B}}$.*

PLP to BN. We turn to the converse direction: starting from a PLP model $[\![-]\!]_{\mathbb{P}} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Stoch}(\mathbf{2})$, we construct a BN model $[\![-]\!]_{\mathbb{B}}$. The main insight is that this construction is purely syntactical: the hurdle to take is figuring out the correct syntax category Σ_H, and a "syntax translation" functor $\mathcal{F} \colon \mathsf{SynBN}_H \to \mathsf{SynLP}_{\mathbb{L}}$ (note the direction!). Then the BN will be the model defined by the composite functor $[\![-]\!]_{\mathbb{P}} \circ \mathcal{F} \colon \mathsf{SynBN}_H \to \mathbf{Stoch}(\mathbf{2})$.

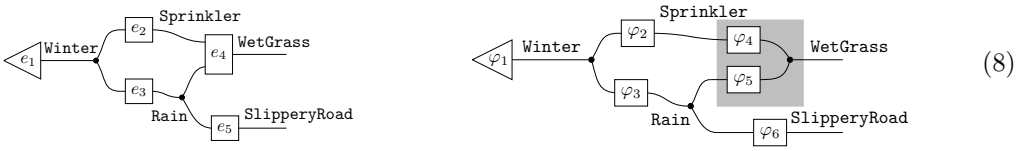
First, we define generating objects V_H and generating morphisms Σ_H yielding our syntax category $\mathsf{SynBN}_H \coloneqq \mathsf{freeCD}(V_H, \Sigma_H)$. We let $V_H = At$, and Σ_H be

$$\left\{ \begin{array}{c} B_1 \\ \vdots \\ B_m \end{array} \boxed{e} \; A \;\middle|\; \{B_1, \dots, B_m\} = \{B \mid \exists \varphi \in \mathbb{L} \text{ such that } A = \mathsf{head}(\varphi), B \in \mathsf{body}(\varphi)\} \right\}$$

Intuitively, inputs of e are obtained by combining all $\mathbb{L}$-clauses with the same head A. We now define the "translation" $\mathcal{F}\colon \mathsf{SynBN}_H \to \mathsf{SynLP}_{\mathbb{L}}$. The idea is that $\mathcal{F}$ "decomposes" children and their parents into $\mathbb{L}$-clauses. Formally, it is the identity on objects and on morphisms it is freely defined by the following mapping on the generating morphisms of SynBN_H:

- For $d \in \{\multimap, \bullet\!\!\rightarrow\}$, $\mathcal{F}(d) \coloneqq d$.
- For each generating morphism $\begin{array}{c} B_1 \\ \vdots \\ B_m \end{array} \boxed{e} \; A$ in Σ_H, $\mathcal{F}(e) \coloneqq comp_A$ (see Definition 14).

▶ **Example 21.** Recall $\mathbb{P}_{\text{wet}}$ from Example 1. Applying $\mathcal{F}_{\text{wet}}\colon \mathsf{SynBN}_{G_{\text{wet}}} \to \mathsf{SynLP}_{\mathbb{P}_{\text{wet}}}$ to the string diagram below left (*cf.* the DAG of Example 18) yields the string diagram below right, where the φ_is are as in Example 3.



(8)

Moreover, the $[\![-]\!]_{\mathbb{P}} \circ \mathcal{F}_{\text{wet}}$-image of the string diagram in the grey block yields precisely the conditional probability distribution $\mathbf{2}_{\texttt{Sprinkler}} \times \mathbf{2}_{\texttt{Rain}} \to \mathcal{D}(\mathbf{2}_{\texttt{WetGrass}})$ in $\mathbb{B}_{\text{wet}}$ represented by the conditional probability distribution in (7).

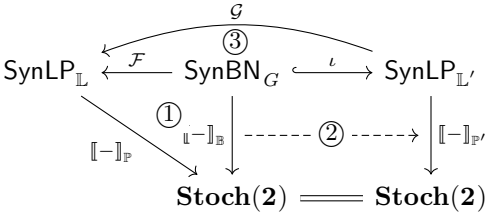
▶ Remark 22. Thanks to the separation of syntax and semantics, our construction is able to simplify and divide in two steps what in the literature (*cf.* [31]) is performed in a single step. In the traditional approach, from $\mathbb{P}$ a DAG is constructed with "AND" nodes and "noisy-OR" nodes [31], from which one derives the conditional probability distributions. Instead, we construct a simpler DAG $H = (V_H, \Sigma_H)$ – in fact, a syntax category SynBN_H encoding H – and only as a next step we introduce a richer structure (modelled with $\bullet\!\!\rightarrow$, see (8)) via $\mathcal{F}_{\text{wet}}$. Moreover, all these steps are performed at a purely syntactic level: obtaining the conditional probabilities is "delegated" to composition with the given functor $[\![-]\!]_{\mathbb{P}} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Stoch}(\mathbf{2})$.

As with the converse direction, we may verify (see Appendix B) that the BN derived from $\mathbb{P}$ in the traditional way (*cf.* [31]) coincides with the one obtained via our construction.

▶ **Proposition 23.** *Let $\mathbb{B} = (G, \Pr)$ be the Bayesian network constructed from $\mathbb{P}$, then $\Sigma_G = \Sigma_H$, and $[\![-]\!]_{\mathbb{B}} = [\![-]\!]_{\mathbb{P}} \circ \mathcal{F}$.*

▶ Remark 24. Note that, unlike Proposition 23, in Proposition 20 we cannot obtain the PLP model $[\![-]\!]_{\mathbb{P}}$ as the composition of the BN model $[\![-]\!]_{\mathbb{B}}$ and a functor between syntax categories. This is because we lack a functor $\mathsf{SynLP}_{\mathbb{L}} \to \mathsf{SynBN}_{\mathbb{B}}$: $\mathsf{SynLP}_{\mathbb{L}}$ has a richer structure than $\mathsf{SynBN}_{\mathbb{B}}$, and there is no canonical way to map the monoid structure $\bullet\!\!\rightarrow, \leftarrow\!\!\bullet$.

The two constructions of this section are summarised by the following commutative diagram:



(9)

Given the definition of SynBN_G and $\mathsf{SynLP}_{\mathbb{L}'}$, we may let $\mathcal{G}$ be $\mathcal{F}$ plus the clause that $\mathcal{G}(d) = d$ for $d \in \{\multimap, \bullet\!\!-\}$. Proposition 23 amounts to commutativity of ①, and ② states that $[\![-]\!]_{\mathbb{B}}$ factors through its CDMU-lifting $[\![-]\!]_{\mathbb{P}'}$ via the inclusion functor ι. ③ connects the program $\mathbb{P}'$ obtained from $\mathbb{B}$ with the original program $\mathbb{P}$ (where $\mathbb{L} = [\mathbb{P}]$ and $\mathbb{L}' = [\mathbb{P}']$).

5 Weighted Logic Programming

We conclude by extending our approach to encompass weighted logic programming (WLP), a generalisation of logic programming for specifying dynamic programming algorithms [14, 10, 15]. We will provide a functorial semantics for WLP, based on which one can express the standard semantics of WLP diagrammatically.

The syntax category is the same as for LP and PLP, reflecting the intuition that the extension provided by WLP only affects the semantics layer. The semantics category for WLP should reflect the fact that atoms are now interpreted no longer as boolean values but as values in a given semiring $\mathcal{K}$. Interestingly, at the intuitive level, the variation required does not change the morphisms of the "basic" semantics category $\mathbf{Set}(\mathbf{2})$, as in the PLP case, but the objects. Whereas the semantics category $\mathbf{Stoch}(\mathbf{2})$ for PLP can be thought as a "Kleisli" variation of $\mathbf{Set}(\mathbf{2})$, the semantics category $\mathbf{Set}(\mathcal{K})$ for WLP simply changes the generating objects of $\mathbf{Set}(\mathbf{2})$ from copies of the Boolean algebra $\mathbf{2}$ to copies of the semiring $\mathcal{K}$.

More precisely, we define $\mathbf{Set}(\mathcal{K})$ as the category whose objects are finite products of the form $\mathcal{K}_{A_1} \times \cdots \times \mathcal{K}_{A_k}$ (*cf. Section 2*). In particular, the singleton set $\mathbf{1}$ is the empty product. The $\mathbf{Set}(\mathcal{K})$-morphisms are functions between the underlying sets. Now, every WLP program $\mathbb{P}$ with atom set At and $\mathbb{L} \coloneqq [\mathbb{P}]$ uniquely determines a functor $[\![-]\!]_{\mathbb{P}} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathcal{K})$. On objects, $[\![A]\!]_{\mathbb{P}} \coloneqq \mathcal{K}_A$ for each $A \in At$. On morphisms, for the CDMU structure we define

$$[\![\multimap_A]\!]_{\mathbb{P}} : \begin{array}{ccc} \mathcal{K}_A & \to & \mathcal{K}_A \times \mathcal{K}_A \\ x & \mapsto & (x, x) \end{array} \quad [\![\bullet\!\!-_A]\!]_{\mathbb{P}} : \begin{array}{ccc} \mathcal{K}_A \times \mathcal{K}_A & \to & \mathcal{K}_A \\ (x, y) & \mapsto & x + y \end{array} \quad [\![\multimap\!\bullet_A]\!]_{\mathbb{P}} : \begin{array}{ccc} \mathcal{K}_A & \to & \mathbf{1} \\ x & \mapsto & * \end{array} \quad [\![\bullet\!\!-\!\bullet_A]\!]_{\mathbb{P}} : \begin{array}{ccc} \mathbf{1} & \to & \mathcal{K}_A \\ * & \mapsto & \mathbf{0}_A \end{array}$$

For each generating morphism $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \; A$ in $\Sigma_{\mathbb{L}}$, $[\![\varphi]\!]_{\mathbb{P}}$ maps $u = (u_1, \ldots, u_m) \in \mathcal{K}_{B_1} \times \cdots \times \mathcal{K}_{B_m}$ to $lab(\psi) \cdot u_1 \cdot \cdots \cdot u_m$, where ψ is the (unique) $\mathbb{P}$-clause satisfying $\tau^{wt}(\psi) = \varphi$. $\mathbf{Set}(\mathcal{K})$ being a CDMU category (Lemma 29, Appendix B) guarantees that $[\![-]\!]_{\mathbb{P}}$ is well-defined.

▶ **Proposition 25.** *There is a 1-1 correspondence between weighted logic programs based on* $\mathbb{L}$ *and generator-preserving CDMU functors* $\mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathcal{K})$.

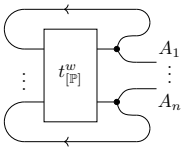
Analogously to what we did with LP and PLP, we conclude the section by describing how the semantics of WLP (see Sec. 2) can be expressed with string diagrams of the syntax category.

Let us fix a WLP $\mathbb{P}$, whose atom set is $At = \{A_1, \ldots, A_n\}$. First, the string diagram $t^w_{[\mathbb{P}]}$ expressing the weighted immediate consequence operator $\mathbf{T}^w_{\mathbb{P}}$ (*cf.* Section 2) is defined as in (4), where $\varphi_1, \ldots, \varphi_N$ are the generating morphisms in $\Sigma_{[\mathbb{P}]}$. As expected, $\left[\!\!\left[t^w_{[\mathbb{P}]} \right]\!\!\right]_{\mathbb{P}} = \mathbf{T}^w_{\mathbb{P}}$.

As a second step, we want to express the semantics of WLP, as defined in Section 2. As this is a least fixed point semantics, we use the extended syntax $\mathsf{SynLP}^{Tr}_{[\mathbb{P}]} = \mathsf{freeTrCDMU}(At, \Sigma_{[\mathbb{P}]})$ from Section 3.3, and interpret it in the category $\mathbf{Rel}(\mathcal{K})$, whose objects are that of $\mathbf{Set}(\mathcal{K})$ and morphisms $A \to B$ are relations $R \subseteq A \times B$. Again, the CDMU structure on $\mathbf{Rel}(\mathcal{K})$ consists of the relations derived from that on $\mathbf{Set}(\mathcal{K})$. The program $\mathbb{P}$ uniquely determines a traced CDMU functor $[\![-]\!]^{Tr}_{\mathbb{P}} : \mathsf{SynLP}^{Tr}_{[\mathbb{P}]} \to \mathbf{Rel}(\mathcal{K})$ inductively defined on the generating morphisms, in a similar manner as that for classical logic program (*cf.* Section 3.3). In particular, the inductive step includes that, for each string diagram $\begin{smallmatrix} X \\ A \end{smallmatrix} \boxed{f} \begin{smallmatrix} X \\ B \end{smallmatrix}$ in $\mathsf{SynLP}^{Tr}_{[\mathbb{P}]}$,

$$\left[\!\!\left[\; \begin{smallmatrix} \circlearrowright \\ A \end{smallmatrix} \boxed{f} \; B \right]\!\!\right]^{Tr}_{\mathbb{P}} = \{(a, b) \in [\![A]\!]^{Tr}_{\mathbb{P}} \times [\![B]\!]^{Tr}_{\mathbb{P}} \mid \exists x \in [\![X]\!]^{Tr}_{\mathbb{P}} : ((x, a), (x, b)) \in [\![f]\!]^{Tr}_{\mathbb{P}}\}.$$

Since atom weights $(weight_{\mathbb{P}}(A_1), \ldots, weight_{\mathbb{P}}(A_n))$ are the least fixed point of $\mathbf{T}^w_{\mathbb{P}}$ (*cf.* Section 2), we can express them as $t^w_{[\mathbb{P}]}$ with "feedback" wires, resulting in the string diagram $W_{[\mathbb{P}]}$ defined as on the right. Formally, this means that $(\min [\![W_{[\mathbb{P}]}]\!]^{Tr}_{\mathbb{P}})(i) = weight_{\mathbb{P}}(A_i)$, for $i = 1, \ldots, n$.



6 Conclusions

In this paper we established a functorial perspective on logic programming, encompassing classical, probabilistic, and weighted programs. This allowed us to propose an original viewpoint on some well-known semantic constructs for these formalisms, and on the correspondence between probabilistic programs and Bayesian networks.
This work paves the way for several future developments:

- The functorial view on the equivalence between probabilistic programs and Bayesian networks provides a basis for a comparative analysis of various inference tasks in logic programming with inference in Bayesian reasoning, for which we may rely on several recent categorical approaches [11, 23, 24, 22, 21]. In particular, the maximum a posteriori (MAP) task [5], most probable explanation (MPE) task [5], and inductive logic programming (ILP) [32] seem mostly promising.
- We would like to extend the equivalence between probabilistic programs and Bayesian networks to richer classes. On the side of logic programming, this includes CP-logic [39] and logic programs with annotated disjunctions (LPAD) [40]. On the side of Bayesian networks, we may study causal diagrams for structural equation models (SEM) [34].
- For simplicity, this paper only deals with the ground case: all the logic programs we consider are propositional, without variables. We leave a functorial semantics of arbitrary logic programs (with variables) as future work, potentially using more sophisticated formalisms of string diagrams, such as nominal diagrams [4].
- Classical, probabilistic and weighted logic programming already attracted a categorical modelling, in terms of coalgebras [27, 26, 7, 6, 19]. A natural research direction is exploring possible synergies between these works and ours. In particular, a string diagrammatic analysis of coinductive logic programming [28, 20] seems particularly intriguing.

References

1 Samson Abramsky. Abstract scalars, loops, and free traced and strongly compact closed categories. In *Algebra and Coalgebra in Computer Science: First International Conference, CALCO 2005, Swansea, UK, September 3-6, 2005, Proceedings*, volume 3629, October 2009. doi:10.1007/11548133_1.

2 Joyal Andre, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447–468, April 1996. doi:10.1017/S0305004100074338.

3 Krzysztof R Apt, Howard A Blair, and Adrian Walker. Towards a theory of declarative knowledge. In *Foundations of deductive databases and logic programming*, pages 89–148. Elsevier, 1988.

4 Samuel Balco and Alexander Kurz. Nominal string diagrams. In Markus Roggenbach and Ana Sokolova, editors, *8th Conference on Algebra and Coalgebra in Computer Science, CALCO 2019, June 3-6, 2019, London, United Kingdom*, volume 139 of *LIPIcs*, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.CALCO.2019.18.

5 Elena Bellodi, Marco Alberti, Fabrizio Riguzzi, and Riccardo Zese. Map inference for probabilistic logic programming. *Theory and Practice of Logic Programming*, 20(5):641–655, 2020.

6 Filippo Bonchi and Fabio Zanasi. Saturated semantics for coalgebraic logic programming. In *Algebra and Coalgebra in Computer Science - 5th International Conference, CALCO 2013, Warsaw, Poland, September 3-6, 2013. Proceedings*, pages 80–94, 2013. doi:10.1007/978-3-642-40206-7_8.

7 Filippo Bonchi and Fabio Zanasi. Bialgebraic semantics for logic programming. *Logical Methods in Computer Science*, 11(1), 2015. doi:10.2168/LMCS-11(1:14)2015.

8 Ivan Bratko. *Prolog programming for artificial intelligence*. Pearson education, 2001.

9 Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic programming and databases*. Springer Science & Business Media, 2012.

10 Shay Cohen, Robert Simmons, and Noah Smith. Products of weighted logic programs. *Computing Research Repository - CORR*, 11, June 2010. doi:10.1017/S1471068410000529.

11 Jared Culbertson and Kirk Sturtz. A categorical foundation for bayesian probability. *Applied Categorical Structures*, 22(4):647–662, August 2013. doi:10.1007/s10485-013-9324-9.

12 Eduardo Menezes de Morais and Marcelo Finger. Probabilistic answer set programming. In *2013 Brazilian Conference on Intelligent Systems*, pages 150–156. IEEE, 2013.

13 Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2468–2473, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL: http://dl.acm.org/citation.cfm?id=1625275.1625673.

14 Didier Dubois, Lluas Godo, and Henri Prade. Weighted logics for artificial intelligence : an introductory discussion. *International Journal of Approximate Reasoning*, 55(9):1819–1829, 2014. Weighted Logics for Artificial Intelligence. doi:10.1016/j.ijar.2014.08.002.

15 Jason Eisner and John Blatz. Program transformations for optimization of parsing algorithms and other weighted logic programs. In Shuly Wintner, editor, *Proceedings of FG 2006: The 11th Conference on Formal Grammar*, pages 45–85. CSLI Publications, 2007. URL: http://cs.jhu.edu/~jason/papers/#eisner-blatz-2007.

16 Brendan Fong. Causal theories: A categorical perspective on bayesian networks. 2013. arXiv:1301.6201.

17 Michael Gelfond. On stratified autoepistemic theories. In *AAAI*, volume 87, pages 207–211, 1987.

18 Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

19 Tao Gu and Fabio Zanasi. Coalgebraic Semantics for Probabilistic Logic Programming. *Logical Methods in Computer Science*, Volume 17, Issue 2, April 2021. URL: https://lmcs.episciences.org/7365.

20 Gopal Gupta, Ajay Bansal, Richard Min, Luke Simon, and Ajay Mallya. Coinductive logic programming and its applications. In Véronica Dahl and Ilkka Niemelä, editors, *Logic Programming*, pages 27–44, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

21 Bart Jacobs. A channel-based exact inference algorithm for bayesian networks. *CoRR*, abs/1804.08032, 2018. arXiv:1804.08032.

22 Bart Jacobs, Aleks Kissinger, and Fabio Zanasi. Causal inference by string diagram surgery. In Mikołaj Bojańczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures*, pages 313–329, Cham, 2019. Springer International Publishing.

23 Bart Jacobs and Fabio Zanasi. A predicate/state transformer semantics for bayesian learning. *Electronic Notes in Theoretical Computer Science*, 325:185–200, 2016. The Thirty-second Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXII). doi:10.1016/j.entcs.2016.09.038.

24 Bart Jacobs and Fabio Zanasi. A Formal Semantics of Influence in Bayesian Reasoning. In Kim G. Larsen, Hans L. Bodlaender, and Jean-Francois Raskin, editors, *42nd International*

Symposium on Mathematical Foundations of Computer Science (MFCS 2017), volume 83 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.MFCS.2017.21.

25 Bart Jacobs and Fabio Zanasi. The logical essentials of bayesian reasoning. *arXiv preprint arXiv:1804.01193*, 2018.

26 Ekaterina Komendantskaya and John Power. Coalgebraic semantics for derivations in logic programming. In *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings*, pages 268–282, 2011. doi:10.1007/978-3-642-22944-2_19.

27 Ekaterina Komendantskaya and John Power. Logic programming: Laxness and saturation. *J. Log. Algebraic Methods Program.*, 101:1–21, 2018. doi:10.1016/j.jlamp.2018.07.004.

28 Ekaterina Komendantskaya, John Power, and Martin Schmidt. Coalgebraic logic programming: from semantics to implementation. *J. Log. Comput.*, 26(2):745–783, 2016. doi:10.1093/logcom/exu026.

29 F William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(5):869, 1963.

30 Vladimir Lifschitz. On the declarative semantics of logic programs with negation. In *Foundations of deductive databases and logic programming*, pages 177–192. Elsevier, 1988.

31 Wannes Meert, Jan Struyf, and Hendrik Blockeel. Learning ground cp-logic theories by leveraging bayesian network learning techniques. *Fundam. Inform.*, 89:131–160, January 2008.

32 Stephen Muggleton and Luc de Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19-20:629–679, 1994. Special Issue: Ten Years of Logic Programming. doi:10.1016/0743-1066(94)90035-3.

33 Ulf Nilsson and Jan Małuszyński. *Logic, programming and Prolog*. Wiley Chichester, 1990.

34 Judea Pearl. *Causality*. Cambridge university press, 2009.

35 Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In *Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming, Tokyo, Japan, June 13-16, 1995*, pages 715–729, 1995.

36 Taisuke Sato and Yoshitaka Kameya. New advances in logic-based probabilistic modeling by prism. In *Probabilistic inductive logic programming*, pages 118–155. Springer, 2008.

37 P. Selinger. *A Survey of Graphical Languages for Monoidal Categories*, pages 289–355. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-12821-9_4.

38 Allen Van Gelder, Kenneth A Ross, and John S Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3):619–649, 1991.

39 Joost Vennekens, Marc Denecker, and Maurice Bruynooghe. Representing causal information about a probabilistic process. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *Logics in Artificial Intelligence*, pages 452–464, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

40 Joost Vennekens, Sofie Verbaeten, and Maurice Bruynooghe. Logic programs with annotated disjunctions. In *Logic Programming*, pages 431–445, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. doi:10.1007/978-3-540-27775-0_30.

A Functorial semantics of Bayesian networks

We briefly recall the functorial semantics for Bayesian networks from [22]. The semantics category is the (CD) category **Stoch** of finite sets and stochastic processes. Given a Bayesian network $\mathbb{B} = (G, \Pr)$, the syntax category SynBN_G the freely generated CD category $\mathsf{freeCD}(V_G, \Sigma_G)$, where $\Sigma_G \coloneqq \left\{ \begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{e} \; A \;\middle|\; A \in V_G \text{ and } pa(A) = \{B_1, \ldots, B_m\} \right\}$. The functorial semantics of Bayesian networks can be precisely stated as:

▶ **Proposition 26** ([22], Proposition 3.1)**.** *There is 1-1 correspondence between Bayesian networks based on a DAG G and CD functors* $\mathsf{SynBN}_G \to \mathbf{Stoch}$.

Proposition 19, the functorial semantics of boolean-valued Bayesian networks, follows immediately from Proposition 26 by restriction to boolean-valued Bayesian networks.

B Omitted proofs

▶ **Lemma 27.** $\mathbf{Set}(\mathbf{2})$ *is a CDMU category.*

Proof. The category $\langle \mathbf{Set}(\mathbf{2}), \times, \mathbf{1}\rangle$ is a SMC category, where $\times$ is the cartesian product in $\mathbf{Set}$, and $\mathbf{1}$ is the singleton set $\{*\}$ (which is the 0-ary product of copies of $\mathbf{2}$). We define the CDMU structure on the two-element boolean algebra $\mathbf{2} = \{0, 1\}$, and that for a copy $\mathbf{2}_A$ follows by replacing 0 and 1 with $\neg A$ and A, respectively:

$$\multimap_{\mathbf{2}}\colon \mathbf{2} \to \mathbf{2}\times\mathbf{2},\ x \mapsto (x,x) \qquad \multimap_{\mathbf{2}}\colon \mathbf{2}\times\mathbf{2} \to \mathbf{2},\ (x,y) \mapsto x \vee y \qquad \bullet\!\!\multimap_{\mathbf{2}}\colon \mathbf{2} \to \mathbf{1},\ x \mapsto * \qquad \bullet\!\!\multimap_{\mathbf{2}}\colon \mathbf{1} \to \mathbf{2},\ * \mapsto 0$$

We only verify the equations for the monoid structure. Given arbitrary $x, y, z \in \mathbf{2}$,

- $((id \otimes \bullet\!\!\multimap); \multimap)(x) = \multimap(x, 0) = x \vee 0 = x$. Similarly $(\bullet\!\!\multimap \otimes id); \multimap = id$.
- $((\multimap \otimes id); \multimap)(x, y, z) = \multimap(x \vee y, z) = x \vee y \vee z = ((id \otimes \multimap); \multimap)(x, y, z)$.
- $(\times; \multimap)(x, y) = \multimap(y, x) = x \vee y = \multimap(x, y)$.

The CDMU structure on arbitrary objects of the form $\mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_k}$ are defined pointwise, and it follows immediately that they satisfy the CDMU equations. ◀

Proof of Proposition 5. The construction of a CDMU functor $[\![-]\!]_{\mathbb{P}}$ from a program $\mathbb{P}$ is already discussion in Subsection 4.1.

For the other direction, given a generator-preserving CDMU functor $\mathcal{G}\colon \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$, we know $\mathcal{G}(A)$ is some copy of $\mathbf{2}$, say $\mathbf{2}_A$, for each $A \in At$. For each generating morphism $\begin{matrix} B_1 \\ \vdots \\ B_m \end{matrix}\!\!\boxed{\varphi}\!\!- A$ in $\Sigma_{\mathbb{L}}$, $\mathcal{G}(\varphi)$ is a function $f\colon \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m} \to \mathbf{2}_A$. Suppose that the support of $\mathcal{G}(\varphi)$ – the inverse image of $A \in \mathbf{2}_A$ under f – is $\{v_1, \ldots, v_d\}$. Then we construct d-many clauses $\varphi_1, \ldots, \varphi_d$ such that $\tau^{gen}(\varphi_j) = A \leftarrow B_1, \ldots, B_m.$, one for each v_i. $v_i(B_{p_1}) = B_{p_1}, \cdots = v_i(B_{p_k}) = B_{p_k}$, $v_i(B_{q_1}) = \neg B_{q_1}, \ldots, v_i(B_{q_\ell}) = \neg B_{q_\ell}$ (with $k + \ell = m$ and $\{p_1, \ldots, p_k, q_1, \ldots, q_\ell\} = \{1, \ldots, m\}$), then let clause φ_i be $A \leftarrow B_{p_1}, \ldots, B_{p_k}, \neg B_{q_1}, \ldots, \neg B_{q_\ell}.$. The program $\langle \mathcal{G}\rangle$ is then the collection of all the clauses constructed from each generating morphism in $\Sigma_{\mathbb{L}}$.

We show that the aforementioned two constructions are inverse to each other.

- We start from a logic program $\mathbb{P}$ based on $\mathsf{SynLP}_{\mathbb{L}}$. For an arbitrary $\varphi \equiv A \leftarrow B_1, \ldots, B_m.$ in $\mathbb{L}$, suppose $\{\varphi_1, \ldots, \varphi_d\}$ are all the clauses φ_j such that $\tau^{gen}(\varphi_j) = \varphi$, then $[\![\varphi]\!]_{\mathbb{P}}$ is defined as a function $\mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m} \to \mathbf{2}_A$ whose support (namely $\mathcal{G}(\varphi)^{-1}(A)$) has size d. Then from $[\![\varphi]\!]_{\mathbb{P}}$ we retrieve d clauses, which are exactly $\varphi_1, \ldots, \varphi_d$: if $v \in \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$ satisfies $[\![\varphi]\!]_{\mathbb{P}}(v) = A$, then there exists some φ_j such that for all $i = 1, \ldots, m$, $v(B_i) = B_i$ if and only if B_i appears positively (namely as B) in $\mathsf{body}(\varphi_j)$, and the clause induced by v is exactly φ_j itself.
- We start from a functor $\mathcal{G}\colon \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$. Fix some $\mathbb{L}$-clause φ, and suppose $\mathcal{G}(\varphi)^{-1}(A) = \{v_1, \ldots, v_d\}$. Then $\mathcal{G}$ determines d-many clauses $\varphi_1, \ldots, \varphi_d$ whose image under τ^{gen} is φ. Then the action of the functor $[\![-]\!]_{\langle \mathcal{G}\rangle} : \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$ on $\varphi \in \mathbb{L}$ is totally determined by $\{\varphi_1, \ldots, \varphi_d\}$: $[\![\varphi]\!]_{\langle \mathcal{G}\rangle}(v) = A$ if and only if v is compatible with $\mathsf{body}(\varphi_j)$ for some $\varphi_j \in \{\varphi_1, \ldots, \varphi_d\}$; but φ_j is exactly represented by v_j, so $[\![\varphi]\!]_{\langle \mathcal{G}\rangle}(v) = A$ if and only if $v = v_j$, for some $v_j \in \{v_1, \ldots, v_d\}$. This means that $[\![\varphi]\!]_{\langle \mathcal{G}\rangle} = \mathcal{G}(\varphi)$, for arbitrary $\mathbb{L}$-clause φ.

Therefore there is a bijection between logic programs based on $\mathbb{L}$ and generator-preserving CDMU functors $\mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Set}(\mathbf{2})$. ◀

Proof of Proposition 7. We are given a logic program $\mathbb{P}$ based on $\mathbb{L}$ with atom set $At = \{A_1, \ldots, A_n\}$. For arbitrary $u \in \mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_n}$ and $A \in At$, we know $\mathbf{T}_{\mathbb{P}}(u)(A) = A$ if and only if there exists $\psi \in \mathbb{P}$ such that $\mathsf{head}(\psi) = A$ and $u \vDash \mathsf{body}(\psi)$. Let $\varphi = \tau^{gen}(\psi)$, then $u \vDash \mathsf{body}(\psi)$ if and only if $[\![\varphi]\!]_{\mathbb{P}} : u|_{\varphi} \mapsto A$, where $u|_{\varphi}$ is the projection of u (as a tuple of values) to the set of atoms in $\mathsf{body}(\varphi)$. So $\mathbf{T}_{\mathbb{P}}(u)(A) = A$ if and only if there exists $\varphi \in [\mathbb{P}]$ such that $\mathsf{head}(\varphi) = A$ and $[\![\varphi]\!]_{\mathbb{P}} : u|_{\varphi} \mapsto A$. By the interpretation of $\multimap$ and $\rightarrowtail$ under $[\![-]\!]_{\mathbb{P}}$, this again is equivalent to that $\left[\!\left[t_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}(u)(A) = A$. ◀

Proof of Proposition 10 . It suffices to show that $\left[\!\left[h_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}^{Tr}$ is exactly the set of all fixed points of $\mathbf{T}_{\mathbb{P}}$. On one hand, suppose u is a fixed point of $\mathbf{T}_{\mathbb{P}}$, then by Proposition 7 $\left[\!\left[t_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}^{Tr}(u) = \mathbf{T}_{\mathbb{P}}(u) = u$. It follows from the definition of $[\![-]\!]_{\mathbb{P}}^{Tr}$ on traced morphisms that u is an element in $\left[\!\left[h_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}^{Tr}$. On the other hand, let v be a state in $\left[\!\left[h_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}^{Tr}$, then by definition of $[\![-]\!]_{\mathbb{P}}^{Tr}$ on traced morphisms, $\left[\!\left[t_{[\mathbb{P}]}\right]\!\right]_{\mathbb{P}}^{Tr}(v) = v$, so $\mathbf{T}_{\mathbb{P}}(v) = v$. ◀

Proof of Proposition 13. The direction from a PLP $\mathbb{P}$ to a generator-preserving CDMU functor $[\![-]\!]_{\mathbb{P}}$ is already discussed in Subsection 4.1.

For the other direction, given a generator-preserving CDMU functor $\mathcal{G} \colon \mathsf{SynLP}_{\mathbb{L}} \to \mathbf{Stoch}(\mathbf{2})$, it maps each $A \in At$ to a copy of $\mathbf{2}$, say $\mathbf{2}_A$. We define a PLP program $\langle \mathcal{G} \rangle$. For an arbitrary $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A$ in $\Sigma_{\mathbb{L}}$, $\mathcal{G}(\varphi)$ is a function $\mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m} \to \mathcal{D}(\mathbf{2}_A)$. Let $\{v_1, \ldots, v_d\}$ be the set of all $v \in \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$ such that $\mathcal{G}(\varphi)(v) \neq 1|\neg A\rangle$. Then we construct one PLP clause for every such v: suppose $v(B_{i_1}) = B_{i_1}, \ldots, v(B_{i_k}) = B_{i_k}$, $v(B_{j_1}) = \neg B_{j_1}, \ldots, v(B_{j_\ell}) = \neg B_{j_\ell}$ be an enumeration of all the components of the m-tuple v, and $\mathcal{G}(\varphi)(v) = p|A\rangle + (1-p)|\neg A\rangle$, then we define a clause $\varphi_v \equiv p :: A \leftarrow B_{i_1}, \ldots, B_{i_k}, \neg B_{j_1}, \ldots, \neg B_{j_\ell}$. The PLP program $\langle \mathcal{G} \rangle$ consists of all the clauses $\{\varphi_{v_1}, \ldots, \varphi_{v_d}\}$ defined as above for each φ in $\Sigma_{\mathbb{L}}$.

The above two procedures $[\![-]\!]_{(\cdot)}$ and $\langle \cdot \rangle$ are inverse to each other. Starting from a $\mathbb{P}$-clause $\psi \equiv p :: A \to B_{i_1}, \ldots, B_{i_k}, \neg B_{j_1}, \ldots, \neg B_{j_\ell}$. with $\tau^{pr}(\psi) = \varphi$, ψ determines the behaviour of the functor $[\![\varphi]\!]_{\mathbb{P}}$ at the unique input v with $v \vDash \mathsf{body}(\psi)$ as $[\![\varphi]\!]_{\mathbb{P}}(v) = p|A\rangle + (1-p)|\neg A\rangle$. This defines a unique clause in $\langle [\![-]\!]_{\mathbb{P}} \rangle$, which is exactly ψ. Starting from a functor $\mathcal{G}$, it generates 2^m clauses in $\langle \mathcal{G} \rangle$ from each $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A$ in $\Sigma_{\mathbb{L}}$, all of whose τ^{pr}-images are $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A$. This behaviour of $[\![-]\!]_{\langle \mathcal{G} \rangle}$ on $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A$ is determined by these 2^m clauses, which is exactly $\mathcal{G}(\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A)$. ◀

Proof of Proposition 17. It suffices to show $[\![f_{At}]\!]_{\mathbb{P}} = \delta_{\mathbb{P}}(At)$, and that for subsets of At follows by observing that using string diagrams, marginal distributions can be obtained from the joint distributions by applying discarders $\multimap\!\bullet$ [25]. We prove $[\![f_{At}]\!]_{\mathbb{P}} = \delta_{\mathbb{P}}(At)$ by induction on the size of At. If $|At| = 1$, then $\mathbb{P}$ can only contain one clause of the form $\psi \equiv p :: A \leftarrow .$, so $[\![f_{At}]\!]_{\mathbb{P}} = p|A\rangle + (1-p)|\neg A\rangle$, which corresponds to that $\pi_{\mathbb{P}}(A) = p$, $\pi_{\mathbb{P}}(\neg A) = 1 - p$.

Now suppose $|At| = n + 1$ and the proposition holds for all programs with atom size $\leq n$. Because $\mathbb{P}$ is acyclic, we can select an atom $A \in At$ such that there is no $\mathbb{P}$-clause φ with $A \in \mathsf{body}(\varphi)$. Let $\mathbb{P}^A$ be the program consisting of all $\mathbb{P}$-clauses with head A, $\mathbb{P}^-$ be $\mathbb{P} \setminus \mathbb{P}^A$, and At^- be $At \setminus \{A\}$. If $\mathbb{P}^A = \varnothing$, then there is no $\mathbb{P}$-clause with head A, $f_{At} = f_{At^-} \otimes \bullet\!\!-_A$. Thus $[\![f_{At}]\!]_{\mathbb{P}} = \sum_{u \in \mathbf{2}_{At^-}} \delta_{\mathbb{P}^-}(u)|u, 0_A\rangle = \delta_{\mathbb{P}}$

So we assume $\mathbb{P}^A \neq \varnothing$. For each interpretation $\mathcal{I}$ of $\mathbb{P}$, we observe that if $A \in \mathcal{I}$, then a sub-program $\mathbb{L} \subseteq |\mathbb{P}|$ satisfies $M_l(\mathbb{L}) = \mathcal{I}$ if and only if $M_l(\mathbb{L}^-) = \mathcal{I}^-$ and there exists $\varphi \in \mathbb{L}^A$ such that $\mathcal{I} \vDash \mathsf{body}(\varphi)$; if $A \notin \mathcal{I}$, then a sub-program $\mathbb{L} \subseteq |\mathbb{P}|$ satisfies $M_l(\mathbb{L}) = \mathcal{I}$

if and only if $M_l(\mathbb{L}^-) = \mathcal{I}^-$ and $\mathcal{I} \not\models \mathsf{body}(\varphi)$ for all $\varphi \in \mathbb{L}^A$. We assume that $comp_A$ is of the form $[B_1, \ldots, B_k] \to [A]$. We can calculate $\delta_{\mathbb{P}}(At)$ using the induction hypothesis $\delta_{\mathbb{P}^-}(At^-)(\mathcal{I}^-) = [\![f_{At^-}]\!]_{\mathbb{P}}(\mathcal{I}^-)$. For example,

$$
\begin{aligned}
&\delta_{\mathbb{P}}(At)(\mathcal{I}^- \cup \{A\}) \\
=&\sum\{\mu_{\mathbb{P}}(\mathbb{L}) \mid \mathbb{L} \subseteq |\mathbb{P}|, M_l(\mathbb{L}) = \mathcal{I}^- \cup \{A\}\} \\
=&\sum\{\mu_{\mathbb{P}^-}(\mathbb{L}^-) \cdot \mu_{\mathbb{P}^A}(\mathbb{L}^A) \mid \mathbb{L}^- \subseteq |\mathbb{P}^-|, \mathbb{L}^A \subseteq |\mathbb{P}^A|, M_l(\mathbb{L}^-) = \mathcal{I}^-, \exists \varphi \in \mathbb{P}^A \text{ s.t. } \mathcal{I}^- \models \mathsf{body}(\varphi)\} \\
=&\sum\{\mu_{\mathbb{P}^-}(\mathbb{L}^-) \cdot [\![comp_A]\!]_{\mathbb{P}}(\mathcal{I}^-|_{B_1,\ldots,B_k})(A) \mid \mathbb{L}^- \subseteq |\mathbb{P}^-|, M_l(\mathbb{L}^-) = \mathcal{I}^-\} \\
=&\delta_{\mathbb{P}^-}(At^-)(\mathcal{I}^-) \cdot [\![comp_A]\!]_{\mathbb{P}}(\mathcal{I}^-|_{B_1,\ldots,B_k})(A) \\
\overset{\text{IH}}{=}&[\![f_{At^-}]\!]_{\mathbb{P}}(\mathcal{I}^-) \cdot [\![comp_A]\!]_{\mathbb{P}}(\mathcal{I}^-|_{B_1,\ldots,B_k})(A) \\
=&[\![f_{At}]\!]_{\mathbb{P}}(\mathcal{I}^- \cup \{A\})
\end{aligned}
$$

Similarly we can show $\delta_{\mathbb{P}}(At)(\mathcal{I}^-) = [\![f_{At}]\!]_{\mathbb{P}}(\mathcal{I}^-)$. Since every interpretation $\mathcal{I}$ for $\mathbb{P}$ can be divided as an interpretation $\mathcal{I}^-$ on $\mathbb{P}^-$ and a subset of $\{A\}$, it follows that $\delta_{\mathbb{P}} = [\![f_{At}]\!]_{\mathbb{P}}$. ◀

Proof of Proposition 20. For each node A, suppose $pa(A) = \{B_1, \ldots, B_m\}$, then there are exactly 2^m clauses ψ_i in $\mathbb{P}$ whose heads are A (possibly some ψ_i has probability label 0), each satisfying $\tau^{npr}(\psi_i) = A \leftarrow B_1, \ldots, B_m$.. By (3), $\Sigma_{[\mathbb{P}]} = \left\{ \begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A \;\middle|\; pa(A) = \{B_1, \ldots, B_m\} \right\} = \Sigma_{\mathbb{L}}$.

To show that $[\![-]\!]_{\mathbb{P}} = [\![-]\!]^b_{\mathbb{B}}$, it suffices to show that they coincide on the generating morphisms $\Sigma_{\mathbb{L}}$. Given arbitrary $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{\varphi} \!\!- A \in \Sigma_{\mathbb{B}}$ and state $u \in \mathbf{2}_{B_1} \times \cdots \times \mathbf{2}_{B_m}$, we show that $[\![\varphi]\!]_{\mathbb{P}}(u) = [\![\varphi]\!]^b_{\mathbb{B}}(u)$. There is exactly one $\mathbb{P}$-clause ψ satisfying both $\tau^{pr}(\psi) = \varphi$ and $u \models \mathsf{body}(\psi)$. Suppose the corresponding interpretation of u is $\{B_{d_1}, \ldots, B_{d_s}\}$, whose complement regarding $\{B_1, \ldots, B_m\}$ is $\{B_{e_1}, \ldots, B_{e_t}\}$, then

$$[\![\varphi]\!]_{\mathbb{P}}(u) \coloneqq lab(\psi) = \Pr(A \mid B_{d_1}, \ldots, B_{d_s}, \neg B_{e_1}, \ldots, \neg B_{e_t})$$

This coincides with the definition of $[\![\varphi]\!]_{\mathbb{B}}$, thus of $[\![\varphi]\!]^b_{\mathbb{B}}$, on u. ◀

▶ **Lemma 28.** $\mathbf{Stoch}(\mathbf{2})$ *is a CDMU category.*

Proof. First of all, $\langle \mathbf{Stoch}(\mathbf{2}), \times, \mathbf{1} \rangle$ forms a SMC [22]. We define the CDMU structure on $\mathbf{2}$, and that for each copy $\mathbf{2}_A$ follows by replacing 1 and 0 with A and $\neg A$, respectively.

$$
\begin{array}{llllll}
\multimap_{\mathbf{2}}\colon & \mathbf{2} \to \mathbf{2} \times \mathbf{2} & \quad \multimap_{\mathbf{2}}\colon & \mathbf{2} \times \mathbf{2} \to \mathbf{2} & \quad \multimap_{\mathbf{2}}\colon & \mathbf{2} \to \mathbf{1} \\
& x \mapsto 1|(x,x)\rangle & & (x,y) \mapsto 1|x \vee y\rangle & & x \mapsto 1|*\rangle \\
\multimap_{\mathbf{2}}\colon & \mathbf{1} \to \mathbf{2} & & & & \\
& * \mapsto 1|0\rangle & & & &
\end{array}
$$

We verify the CDMU equations for the monoid structure, and that for the comonoid structure can be found in [22]. Given arbitrary $x, y, z \in \mathbf{2}$,

- $((id \otimes \multimap); \multimap)(x) = 1|x \vee 0\rangle = 1|x\rangle$. Similarly $(\multimap \otimes id); \multimap = id$.
- $((\multimap \otimes id); \multimap)(x,y,z) = 1|x \vee y \vee z\rangle = ((id \otimes \multimap); \multimap)(x,y,z)$.
- $(\times; \multimap)(x,y) = 1|x \vee y\rangle = \multimap(x,y)$.

The CDMU structure on arbitrary objects of the form $\mathbf{2}_{A_1} \times \cdots \times \mathbf{2}_{A_k}$ is defined pointwise, and it follows immediately that the structure satisfies CDMU equations. ◀

Proof of Proposition 23. The equivalence of the set of generating morphisms follows immediately from their definitions. To show that $[\![-]\!]_{\mathbb{B}} = [\![-]\!]_{\mathbb{P}} \circ \mathcal{F}$, it suffices to show that for arbitrary generating morphism $\begin{smallmatrix} B_1 \\ \vdots \\ B_m \end{smallmatrix} \boxed{e} \!\!- A$ in Σ_G and state $u = (u_1, \ldots, u_m)$ of $B_1, \ldots, B_m$, $[\![\mathcal{F}(e)]\!]_{\mathbb{P}}(u)$ is equivalent to $\Pr(A = 1 \mid B_1 = x_1, \ldots, B_m = x_m)$.

Suppose $\varphi_1, \ldots, \varphi_k$ are all the $[\mathbb{P}]$-clauses with heads A, so they exhaust all the components of $comp_A$ besides the CDMU structure. For each φ_i of the form $A \leftarrow B_{i_1}, \ldots, B_{i_{m_i}}$, $[\![\varphi_i]\!]_{\mathbb{P}}(u) = p|A\rangle + (1-p)|\neg A\rangle$, where p is the probability label of the $\mathbb{P}$-clause ψ satisfying $[\psi] = \varphi$ and $u \vDash \mathsf{body}(\psi)$, and p is 0 if no such clause exists. Note that if such ψ exists, then it is necessarily unique, so this is well-defined.

Now, suppose we have φ_1, φ_2 with disjoint bodies, and $[\![\varphi_j]\!]_{\mathbb{P}}(u_j) = p_j|A\rangle + (1-p_j)|\neg A\rangle$, where $u_j = u|_{\mathsf{body}(\varphi_j)}$, $j \in \{1,2\}$. We compute $[\![(\varphi_i \otimes \varphi_j); \succ\!\bullet\!-_A]\!]_{\mathbb{P}}(u')$, where $u' = u|_{\mathsf{body}(\varphi_1) \cup \mathsf{body}(\varphi_2)}$. By the definition of $[\![\succ\!\bullet\!-]\!]_{\mathbb{P}}$ and morphisms composition in $\mathbf{Stoch}(\mathbf{2})$, $[\![(\varphi_i \otimes \varphi_j); \succ\!\bullet\!-_A]\!]_{\mathbb{P}}(u') = (1-(1-p_1)(1-p_2))|A\rangle + (1-p_1)(1-p_2)|\neg A\rangle$. Then an induction on the number k of components in $comp_A$ shows that, if $[\![\varphi_j]\!]_{\mathbb{P}}(u|_{\mathsf{body}(\varphi_j)}) = p_j|A\rangle + (1-p_j)|\neg A\rangle$, then $[\![comp_A]\!]_{\mathbb{P}}(u) = (1 - \prod_{j=1}^{k}(1-p_j))|A\rangle + \prod_{j=1}^{k}(1-p_j)|\neg A\rangle$. This is exactly the definition of $\Pr(A \mid \{B_i\}_{i=1}^{k} = u)$ in $\mathbb{B}$. ◀

▶ **Lemma 29.** *The category* $\mathbf{Set}(\mathcal{K})$ *is a CDMU category.*

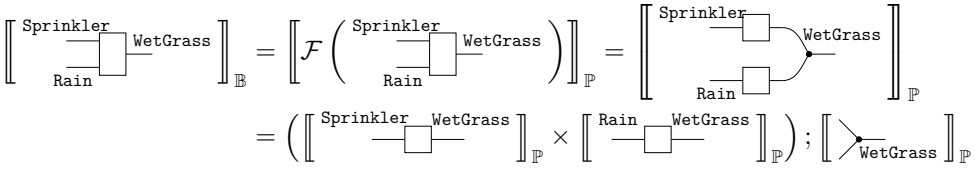
Proof. Category $\mathbf{Set}(\mathcal{K})$ with cartesian product $\times$ and the singleton set $\mathbf{1}$ forms a SMC. Again $\mathbf{1}$ is the 0-ary product of copies of $\mathcal{K}$, thus already included in the definition of $\mathbf{Set}(\mathcal{K})$-objects. We define the CDMU on $\mathcal{K}$, and that on each copy $\mathcal{K}_A$ follows immediately.

$$\begin{array}{llll} \bullet\!-\!\prec_{\mathcal{K}}\colon & \mathcal{K} \to \mathcal{K} \times \mathcal{K} & \succ\!\bullet\!-_{\mathcal{K}}\colon & \mathcal{K} \times \mathcal{K} \to \mathcal{K} \\ & x \mapsto (x,x) & & (x,y) \mapsto x+y \end{array} \qquad \begin{array}{llll} \bullet\!-_{\mathcal{K}}\colon & \mathcal{K} \to \mathbf{1} & \bullet\!-\!\bullet_{\mathcal{K}}\colon & \mathbf{1} \to \mathcal{K} \\ & x \mapsto 1|*\rangle & & * \mapsto \mathbf{0} \end{array}$$

We verify the CDMU equations for the monoid structure, and that for the comonoid structure can be found in [22]. Given arbitrary $x, y, z \in \mathcal{K}$,

- $((id \otimes \bullet\!-\!\prec); \succ\!\bullet\!-)(x) = \succ\!\bullet\!-(x, \mathbf{0}) = x + \mathbf{0} = x$. Similarly $(\bullet\!-\!\prec \otimes id); \succ\!\bullet\!- = id$.
- $((\succ\!\bullet\!- \otimes id); \succ\!\bullet\!-)(x,y,z) = \succ\!\bullet\!-(x+y, z) = (x+y) + z = ((id \otimes \succ\!\bullet\!-); \succ\!\bullet\!-)(x,y,z)$.
- $(\chi; \succ\!\bullet\!-)(x,y) = \succ\!\bullet\!-(y,x) = x + y = y + x = \succ\!\bullet\!-(x,y)$.
- $(\bullet\!-\!\prec; (id \otimes \bullet\!-\!\bullet))(x) = (id \otimes \bullet\!-\!\bullet)(x,x) = x$. Similarly $(\bullet\!-\!\prec; (\bullet\!-\!\bullet \otimes id)) = id$.
- $(\bullet\!-\!\prec; (\bullet\!-\!\prec \otimes id))(x) = (x,x,x) = \bullet\!-\!\prec; (id \otimes \bullet\!-\!\prec)(x)$.
- $(\bullet\!-\!\prec; \chi)(x) = \chi(x,x) = (x,x) = \bullet\!-\!\prec(x)$.

The CDMU structure and verification of CDMU equations for general objects of the form $\mathcal{K}_{A_1} \times \mathcal{K}_{A_k}$ follows immediately from a pointwise definition. ◀



$(0_{\texttt{Sprinkler}}, 1_{\texttt{Rain}}) \mapsto 0.8|1_{\texttt{WetGrass}}\rangle + 0.2|0_{\texttt{WetGrass}}\rangle$

The Central Valuations Monad

Xiaodong Jia
School of Mathematics, Hunan University, Changsha, China

Michael Mislove
Department of Computer Science, Tulane University, New Orleans, LA, USA

Vladimir Zamdzhiev
Université de Lorraine, CNRS, Inria, LORIA, F 54000 Nancy, France

Abstract

We give a commutative valuations monad $\mathcal{Z}$ on the category **DCPO** of dcpo's and Scott-continuous functions. Compared to the commutative valuations monads given in [2], our new monad $\mathcal{Z}$ is larger and it contains all push-forward images of valuations on the unit interval $[0,1]$ along lower semi-continuous maps. We believe that this new monad will be useful in giving domain-theoretic denotational semantics for statistical programming languages with continuous probabilistic choice.

2012 ACM Subject Classification Theory of computation → Denotational semantics

Keywords and phrases Valuations, Commutative Monad, DCPO, Probabilistic Choice, Recursion

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.18

Category Early Ideas

Acknowledgements We thank the anonymous reviewers for their feedback which led to improvements of this paper. Xiaodong Jia acknowledges the support of NSFC (No. 12001181).

1 Introduction

The valuations monad $\mathcal{V}$ on the category **DCPO** of dcpo's and Scott-continuous functions is a staple of the domain-theoretic approach for denotational semantics of programming languages with probabilistic choice and recursion [3, 4]. For a dcpo D, $\mathcal{V}D$ consists of *subprobability valuations* on D, which are the Scott-continuous functions ν from the set σD of Scott open subsets of D to $[0,1]$ satisfying *strictness* ($\nu(\emptyset) = 0$) and *modularity* ($\nu(U) + \nu(V) = \nu(U \cup V) + \nu(U \cap V)$). The set $\mathcal{V}D$ is a dcpo in the *stochastic order*: $\nu_1 \leq \nu_2$ if and only if $\nu_1(U) \leq \nu_2(U)$ for all $U \in \sigma D$. The *unit* of $\mathcal{V}$ at dcpo D is the map $\eta_D \colon D \to \mathcal{V}D :: x \mapsto \delta_x$, where δ_x is the *Dirac valuation* at x, defined by $\delta_x(U) = 1$ if $x \in U$ and $\delta_x(U) = 0$ otherwise. For a Scott-continuous map $f \colon D \to \mathcal{V}E$, the *Kleisli extension* $f^\dagger$ of f is defined by $f^\dagger(\nu)(U) = \int_{x \in X} f(x)(U) d\nu$ for $\nu \in \mathcal{V}D$ and $U \in \sigma E$. The integral in this definition is a Choquet type integral: for a general Scott-continuous function $h \colon D \to [0,1]$, the value of $\int_{x \in X} h d\nu$ is defined to be the Riemann integral $\int_0^1 \nu(h^{-1}(t,1]) dt$. Following this, the action of $\mathcal{V}$ on a Scott-continuous function $g \colon D \to E$ between dcpo's D and E is $\mathcal{V}(g) \stackrel{\text{def}}{=} (\eta_E \circ g)^\dagger$; concretely, for $\nu \in \mathcal{V}D$ and $U \in \sigma E$, $\mathcal{V}(g)(\nu)(U) = \nu(g^{-1}(U))$. Subprobability valuations on general topological spaces and the corresponding integral of lower semi-continuous functions against subprobability valuations can be defined similarly [3].

While it is well-known that $\mathcal{V}$ can be restricted to a commutative monad on the category **DOM** of domains and Scott-continuous functions, it is unknown whether $\mathcal{V}$ can be restricted to any Cartesian closed full subcategory of **DOM**. This is known as the *Jung-Tix problem* [5].

One may note that the category **DCPO** itself is Cartesian closed and $\mathcal{V}$ is a monad on it. What does one lose if we use the category **DCPO** and monad $\mathcal{V}$ for semantics? A short answer is that compared to **DOM**, $\mathcal{V}$ is not known to be *commutative* over **DCPO**, which is an important property for the denotational semantics of programming languages.

Commutativity of $\mathcal{V}$ over **DCPO** is equivalent to showing the following Fubini-style equation

$$\int_{x\in D}\int_{y\in E} h(x,y)d\mu d\nu = \int_{y\in E}\int_{x\in D} h(x,y)d\nu d\mu \tag{1}$$

holds for all dcpo's D and E, all Scott-continuous functions $h\colon D\times E\to[0,1]$ and all $\nu\in\mathcal{V}D, \mu\in\mathcal{V}E$. As pointed out in [2], the main difficulty in establishing (1) over **DCPO** is that the Scott topology on the product dcpo $D\times E$ may be different from the product topology $\sigma D\times\sigma E$. Actually, we do know that Equation (1) holds for those functions h that are continuous when $D\times E$ is given the product topology $\sigma D\times\sigma E$ ([4, Lemma 2.37]).

Instead of directly proving (1), we showed (together with Lindenhovius) how to construct three submonads of $\mathcal{V}$ that are commutative on **DCPO**, and used each one to give a sound and (strongly) adequate semantics to `PFPC` (Probabilistic FixPoint Calculus) [2]. The simplest of those three monads is the monad $\mathcal{M}$. For each dcpo D, $\mathcal{M}D$ is defined to be the smallest sub-dcpo of $\mathcal{V}D$ that contains $\mathcal{S}D$, the family of *simple valuations* on D, where a simple valuation is a finite convex sum of Dirac valuations. The other two commutative monads are denoted $\mathcal{W}$ and $\mathcal{P}$ and the following inclusions hold for each dcpo $D: \mathcal{S}D\subseteq\mathcal{M}D\subseteq\mathcal{W}D\subseteq\mathcal{P}D\subseteq\mathcal{V}D$.

Each of our three monads is large enough to interpret *discrete* probabilistic choice in `PFPC` [2]. However, it is unclear if any of these monads is large enough to interpret *continuous* probabilistic choice. In this note, we define a new commutative valuations monad $\mathcal{Z}$ on the category **DCPO** which is larger than $\mathcal{M},\mathcal{W}$ and $\mathcal{P}$ with the hope of addressing this problem.

2 Central Valuations

Our idea for defining $\mathcal{Z}$ is inspired by the notion of centre in group theory (which always forms an abelian subgroup) and the notion of centre of a premonoidal category (which always forms a monoidal subcategory) [6].

▶ **Definition 1.** *A subprobability valuation ν on a dcpo D is called a* central valuation *if for any dcpo E, any valuation μ on E, and any Scott-continuous function $h\colon D\times E\to[0,1]$, we have*

$$\int_{x\in D}\int_{y\in E} h(x,y)d\mu d\nu = \int_{y\in E}\int_{x\in D} h(x,y)d\nu d\mu.$$

We shall write $\mathcal{Z}D$ for the set of all central valuations on a dcpo D.

It is easy to see that simple valuations are central, and that the central valuations are closed under directed suprema under the stochastic order. Thus, for each dcpo D, $\mathcal{Z}D$ is a sub-dcpo of $\mathcal{V}D$ containing $\mathcal{S}D$. Moreover, we have the following theorem, which can be proved using the *disintegration formula* in [1].

▶ **Theorem 2.** *The assignment $\mathcal{Z}(-)$ extends to a commutative monad over the category* **DCPO** *when equipped with the (co)restricted monad operations of $\mathcal{V}$. In other words, $\mathcal{Z}$ is a commutative submonad of $\mathcal{V}$.*

Proof. The unit of $\mathcal{Z}$ at dcpo D sends each $x\in D$ to δ_x which is obviously a central valuation.

Let $f\colon C\to\mathcal{Z}D$ be a Scott-continuous function. Then f can also be viewed as a Scott-continuous map from C to $\mathcal{V}D$, since $\mathcal{Z}D$ is a sub-dcpo of $\mathcal{V}D$. We prove that $f^\dagger\colon\mathcal{V}C\to\mathcal{V}D$ maps central valuations on C to central valuations on D. Towards this end, we pick μ

from $\mathcal{Z}C$, and assume that E is a dcpo, ν is an arbitrary subprobability valuation on E and $h\colon D \times E \to [0,1]$ is a Scott-continuous map. Then by the disintegration formula (see Lemma 3.1(iii) in [1]) we have that

$$\int_{y\in E}\int_{x\in D} h(x,y)d(f^\dagger(\mu))d\nu = \int_{y\in E}\int_{t\in C}\int_{x\in D} h(x,y)df(t)d\mu d\nu,$$

and the right side of the equation is equal to

$$\int_{t\in C}\int_{x\in D}\int_{y\in E} h(x,y)d\nu df(t)d\mu$$

by the fact that $f(t), t \in D$ and μ are central valuations. Again, by the disintegration formula that is just $\int_{x\in D}\int_{y\in E} h(x,y)d\nu df^\dagger(\mu)$. Hence we have proved that $f^\dagger(\mu)$ is indeed a central valuation provided that μ is. Similar arguments show that the monadic strength also (co)restricts as required. The corresponding (co)restrictions of the monadic operations of $\mathcal{V}$ to $\mathcal{Z}$ validate that $\mathcal{Z}$ is a strong monad on **DCPO**. The commutativity of $\mathcal{Z}$, which is equivalent to Equation (1) holding for all dcpo's D and E and central valuations μ and ν on them, is then obvious by definition of $\mathcal{Z}$. ◀

In fact, it is proved in [2] that all *point-continuous valuations* are central and therefore $\mathcal{S}D \subseteq \mathcal{M}D \subseteq \mathcal{W}D \subseteq \mathcal{P}D \subseteq \mathcal{Z}D \subseteq \mathcal{V}D$ for each dcpo D. Therefore $\mathcal{Z}$ is the largest commutative submonad of $\mathcal{V}$ known so far. Furthermore, observe that $\mathcal{Z} = \mathcal{V}$ iff $\mathcal{V}$ is a commutative monad on **DCPO**. The latter has been an open problem since 1989, and our simple observation leads us to believe $\mathcal{Z}$ is a very large commutative submonad of $\mathcal{V}$.

It is not difficult to see that, in order to model sampling against continuous probability distributions on the interval $[0,1]$, the monad used for the semantics should at least contain the push-forward images of the Lebesgue valuation on $[0,1]$ (equipped with the metric topology) along lower semi-continuous maps. We can demonstrate even more is true of our new monad $\mathcal{Z}$ (see Theorem 4 below). For this, let us first recall that a space X is called *core-compact* if the set $\mathcal{O}X$ of all open subsets of X is a continuous lattice in the inclusion order. Equivalently, X is core-compact if and only if for each open subset U of X and $x \in U$, there exists an open subset V such that $x \in V \ll U$, where $V \ll U$ means that V is *way-below* U in the sense of domain theory. Many important spaces are core-compact. For example, each locally compact space is core-compact, and in particular, the unit interval with the usual topology is compact Hausdorff, hence locally compact hence core-compact.

▶ **Lemma 3.** *Let X be a core-compact topological space. Let D, E be dcpos, and $f\colon X \to D$ a lower semi-continuous map, i.e., f is continuous when D is equipped with the Scott topology. Then the map $f \times \mathrm{id}_E\colon X \times \Sigma E \to D \times E$ is also lower semi-continuous, where ΣE denotes the topological space $(E, \sigma E)$ and $X \times \Sigma E$ is the topological product of X and ΣE.*

Proof. First, we assume that X is core-compact and prove that $f \times \mathrm{id}_E$ is lower semi-continuous. Towards this end, we pick a Scott open subset O of $D \times E$, and assume that $f \times \mathrm{id}_Y((x_0, e_0)) \in O$, that is $(f(x_0), e_0) \in O$. We must find an open neighbourhood U of x_0 in X and a Scott open neighbourhood V of e_0 in E such that $f \times \mathrm{id}_Y(U \times V) \subseteq O$. We let $A = \{x \in X \mid (f(x), e_0) \in O\}$. Then A is just $f^{-1}(O_{e_0})$, where $O_{e_0} = \{d \in D \mid (d, e_0) \in O\}$. Since $f\colon X \to D$ is lower semi-continuous and O_{e_0} is Scott open in D, we know that A is an open neighbourhood of x_0 in X. Now the core-compactness of X enables us to find, in X, an open subset U and a sequence of open subsets $U_i, i \in \mathbb{N}$ such that $x_0 \in U \ll \cdots \ll U_n \cdots \ll U_1 \ll A$. For each $U_n, n \in \mathbb{N}$, we define $V_n = \{e \mid f(x, e) \in O \text{ for all } x \in U_n\}$ and

let $V = \bigcup_{n\in\mathbb{N}} V_n$. Since for each $n \in \mathbb{N}$, $U_n \subseteq A$, we have for all $x \in U_n$, $(f(x), e_0) \in O$. Thus we know that $e_0 \in V_n$ for each $n \in \mathbb{N}$, and hence $e_0 \in V$. Moreover, for any $(x, e) \in U \times V$, there exists a natural number n such that $e \in V_n$, then it follows that $f \times \mathrm{id}_E((x, e)) = (f(x), e) \in f(U) \times V_n \subseteq f(U_n) \times V_n \subseteq O$. The last inclusion is due to the construction of V_n. To sum up, it is true that $f \times \mathrm{id}_E(U \times V) \subseteq O$. Since U is an open subset of X which contains x_0 and $e_0 \in V$, we finish the proof by showing that V is Scott open in E. To this end we let $\{e_i\}_{i\in I}$ be a directed subset of E with $\sup_{i\in I} e_i \in V$. For each $i \in I$, set $W_i = \{x \in X \mid (f(x), e_i) \in O\}$. It is easy to see that $\{W_i \mid i \in I\}$ is a directed family of open subsets of X. Since $\sup_{i\in I} e_i \in V = \bigcup_{n\in\mathbb{N}} V_n$, $\sup_{i\in I} e_i$ is in some V_n. This means that for each $x \in U_n$, $(f(x), \sup_{i\in I} e_i) \in O$. Because O is Scott open, for each $x \in U_n$, there exists $i \in I$ such that $(f(x), e_i) \in O$, i.e., $x \in W_i$. Hence we have that $\sup_{i\in I} e_i \in V_n \subseteq \bigcup_{i\in I} W_i$. Remember that $U_{n+1} \ll U_n$, it follows that $U_{n+1} \subseteq W_j$ for some $j \in I$. By definition of W_j, we know that $f(U_{n+1}) \times \{e_j\} \subseteq O$, which means that $e_j \in V_{n+1}$, this time by definition of V_{n+1}. So we find $j \in I$ with $e_j \in V_{n+1} \subseteq V$, and indeed V is Scott open in E. ◀

▶ **Theorem 4.** *Let X be a core-compact space and f be a lower semi-continuous map from X to a dcpo D. If ν is a valuation on X, then $f_*(\nu) \stackrel{def}{=} \lambda O \in \sigma D.\nu(f^{-1}(O))$, the* push-forward valuation along f*, is a central valuation on D. In particular, for a core-compact dcpo D, all valuations on D are central, i.e., $\mathcal{V}D = \mathcal{Z}D$.*

Proof. By definition, we prove for any dcpo E, continuous valuations μ on E and Scott-continuous map $h\colon D \times E \to [0, 1]$ the equation

$$\int_{x\in D}\int_{y\in E} h(x, y)d\mu df_*(\nu) = \int_{y\in E}\int_{x\in D} h(x, y)df_*(\nu)d\mu$$

holds.

Note that for each $y \in E$, the map $g \stackrel{\text{def}}{=} (x \mapsto \int_{y\in E} h(x, y)d\mu)\colon D \to [0, 1]$ is Scott-continuous, and $f\colon X \to D$ is lower semi-continuous. Hence for the left side of the above equation we have

$$\int_{x\in D}\int_{y\in E} h(x, y)d\mu df_*(\nu) = \int_{x\in X} g(f(x))d\nu = \int_{x\in X}\int_{y\in E} h(f(x), y)d\mu d\nu.$$

The first equality follows form the so-called *change-of-variable* formula, which can be found in [4]. As a consequence of it, we also have that

$$\int_{y\in E}\int_{x\in D} h(x, y)df_*(\nu)d\mu = \int_{y\in E}\int_{x\in X} h(f(x), y)d\nu d\mu.$$

Since X is core-compact and the function $f\colon X \to D$ is lower semi-continuous, by Lemma 3 we know that $f \times \mathrm{id}_E\colon X \times \Sigma E \to X \times Y$ is lower semi-continuous. This implies that the map $(x, y) \mapsto h(f(x), y)\colon X \times \Sigma E \to [0, 1]$ is lower semi-continuous. Hence by Lemma 2.37 in [4] we know that $\int_{x\in X}\int_{y\in E} h(f(x), y)d\mu d\nu = \int_{y\in E}\int_{x\in X} h(f(x), y)d\nu d\mu$, which finishes the proof.

The second claim is a straightforward consequence of the first one. ◀

▶ **Theorem 5.** *Let $f : [0, 1] \to D$ be a lower semi-continuous map into a dcpo D. If ν is any continuous valuation on $[0, 1]$, then $f_*(\nu)$ is a central valuation on D.*

Proof. Since $[0, 1]$ is core-compact in the usual topology, the result follows from Theorem 4. ◀

We have not been able to establish the above theorem for any of the monads $\mathcal{M}, \mathcal{W}$ or $\mathcal{P}$, so we believe that $\mathcal{Z}$ is a promising candidate for modeling *continuous* probabilistic choice. We plan to address this in future work.

References

1 Jean Goubault-Larrecq, Xiaodong Jia, and Clément Théron. A Domain-Theoretic Approach to Statistical Programming Languages, 2021. Preprint. URL: https://arxiv.org/abs/2106.16190.

2 Xiaodong Jia, Bert Lindenhovius, Michael Mislove, and Vladimir Zamdzhiev. Commutative monads for probabilistic programming languages. In *Logic in Computer Science (LICS 2021)*, 2021. arXiv:2102.00510.

3 C. Jones and Gordon D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 186–195. IEEE Computer Society, 1989. doi:10.1109/LICS.1989.39173.

4 Claire Jones. *Probabilistic Non-determinism*. PhD thesis, University of Edinburgh, UK, 1990. URL: http://hdl.handle.net/1842/413.

5 Achim Jung and Regina Tix. The troublesome probabilistic power domain. In *Comprox III, Third Workshop on Computation and Approximation*, volume 13, pages 70–91, 1998.

6 John Power and Edmund Robinson. Premonoidal Categories and Notions of Computation. *Math. Struct. Comput. Sci.*, 7(5):453–468, 1997. doi:10.1017/S0960129597002375.

Coderelictions for Free Exponential Modalities

Jean-Simon Pacaud Lemay
Mathematics and Computer Science Department, Mount Allison University, Sackville, Canada

Abstract

In a categorical model of the multiplicative and exponential fragments of intuitionistic linear logic (MELL), the exponential modality is interpreted as a comonad ! such that each cofree !-coalgebra $!A$ comes equipped with a natural cocommutative comonoid structure. An important case is when ! is a free exponential modality so that $!A$ is the cofree cocommutative comonoid over A. A categorical model of MELL with a free exponential modality is called a Lafont category. A categorical model of differential linear logic is called a differential category, where the differential structure can equivalently be described by a deriving transformation $!A \otimes A \xrightarrow{\mathsf{d}_A} !A$ or a codereliction $A \xrightarrow{\eta_A} !A$. Blute, Lucyshyn-Wright, and O'Neill showed that every Lafont category with finite biproducts is a differential category. However, from a differential linear logic perspective, Blute, Lucyshyn-Wright, and O'Neill's approach is not the usual one since the result was stated in the dual setting and the proof is in terms of the deriving transformation d. In differential linear logic, it is often the codereliction η that is preferred and that plays a more prominent role. In this paper, we provide an alternative proof that every Lafont category (with finite biproducts) is a differential category, where we construct the codereliction η using the couniversal property of the cofree cocommtuative comonoid $!A$ and show that η is unique. To achieve this, we introduce the notion of an infinitesimal augmentation $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$, which in particular is a !-coalgebra and a comonoid morphism, and show that infinitesimal augmentations are in bijective correspondence to coderelictions (and deriving transformations). As such, infinitesimal augmentations provide a new equivalent axiomatization for differential categories in terms of more commonly known concepts. For a free exponential modality, its infinitesimal augmentation is easy to construct and allows one to clearly see the differential structure of a Lafont category, regardless of the construction of $!A$.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Linear logic

Keywords and phrases Differential Categories, Coderelictions, Differential Linear Logic, Free Exponential Modalities, Lafont Categories, Infinitesimal Augmentations

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.19

Funding *Jean-Simon Pacaud Lemay*: The author is financially supported by a NSERC Postdoctoral Fellowship (PDF) - Award #: 456414649

Acknowledgements The author would like to thank the anonymous referees for their comments and suggestions which helped improve this paper, as well as Robert A. G. Seely, Paul-André Melliès, Rory Lucyshyn-Wright for useful discussions, support of this project, and editorial comments.

1 Introduction

In the multiplicative and exponential fragments of intuitionistic linear logic (MELL) [16, 17], the exponential modality !, read as either "of course" or "bang", admitts four structural rules: promotion, dereliction, contraction, and weakening. A categorical model of MELL [2, 25, 26, 29], often called a linear category, is a symmetric monoidal closed category equipped with a **monoidal coalgebra modality** ! [3, 4] which interprets the exponential modality. Briefly, a monoidal coalgebra modality is a symmetric monoidal comonad, capturing the promotion and dereliction rules, such that for each object A, the cofree !-coalgebra $!A$ comes equipped with a natural cocommutative comonoid structure, capturing the contraction and weakening rules.

As shown by Lafont in his Ph.D. thesis [19], an important source of examples of monoidal coalgebra modalities are those for which $!A$ is also the cofree cocommutative comonoid over A. Monoidal coalgebra modalities with this extra couniversal property on $!A$ are known as **free exponential modalities** [27] and models of linear logic with free exponential modalities are known as **Lafont categories** [26]. While free exponential modalities have been around since the beginning with Girard's free exponential modality for coherence spaces [16], new free exponential modalities are still being constructed and studied [11, 20, 22, 31], which shows the importance of these kinds of models. In fact, Lafont categories are arguably the most common example of categorical models of MELL. The simplest construction of a free exponential modality is the one obtained by taking the infinite product of all the symmetrized tensor powers of an object. Melliès, Tabareau, and Tasson give a more general construction [27] as a sequential limit of the symmetrized tensor powers of cofree copointed objects. However, not every free exponential modality can be constructed in these ways. For example, the free exponential modality on the category of modules over an arbitrary commutative (semi)ring R is given by cofree cocommutative R-coalgebras, which are often not simple to describe, but their existence and constructions have been well-studied [1, 28, 32].

Differential linear logic [13, 14, 12], as introduced by Ehrhard and Regnier, is an extension of linear logic which includes a differentiation inference rule, as well as a cocontraction, coweaking, and codereliction for the exponential modality. Blute, Cockett, and Seely then introduced differential categories [6], which were the appropriate categorical structure for modelling differential linear logic. A differential category is an additive symmetric monoidal category with a coalgebra modality which comes equipped with a natural transformation $!A \otimes A \xrightarrow{\mathsf{d}_A} !A$, called a **deriving transformation**, satisfying certain equations based on the properties of differentiation from calculus, such as the Leibniz rule (also known as the product rule) and the chain rule. It is important to note that the basic structure of a differential category is weaker than that of a model of linear logic: the base symmetric monoidal category is not assumed to be closed nor to have finite products, and one only requires a coalgebra modality, which drops the requirement that the underlying comonad be symmetric monoidal. For a monoidal coalgebra modality, differential structure can alternatively be axiomatized in terms of a natural transformation $A \xrightarrow{\eta_A} !A$ called a **codereliction** [6, 3], which is also equivalent to Fiore's notion of a **creation map** [15]. Thus for a monoidal coalgebra modality, there is a bijective correspondence between coderelictions and deriving transformations.

There are many examples of differential categories whose coalgebra modality is a free exponential modality. Indeed, Blute, Cockett, and Seely's original examples of differential categories found in [6] were the category of sets and relations, where the free exponential modality is induced by finite multisets, and the category of vector spaces over an arbitrary field, where the free exponential modality is induced by free symmetric algebras [23]. In [21], Laird, Manzonetto, and McCusker use the dual of the free symmetric algebra to construct a variety of differential categories related to game theory. In [8], Clift and Murfet study the category of vector spaces over an algebraically closed field of characteristic 0 as a categorical model of differential linear logic and uses the fact that the free exponential modality in this case admits a very elegant construction. This raises the natural question of whether free exponential modalities (in an appropriate setting) always comes equipped with a codereliction/deriving transformation, and if a Lafont category is always a differential category. The answer is yes! In [5], Blute, Lucyshyn-Wright, and O'Neill showed that, in the presence of finite biproduct, every free exponential modality admits a deriving transformation, and thus every Lafont category with finite biproducts is a differential category. However, from a differential linear logic perspective, Blute, Lucyshyn-Wright, and O'Neill's approach

is not the usual one since: (a) the result was stated in the dual setting, and (b) the proof and construction involve the deriving transformation rather than the codereliction. The latter reason is important since in differential linear logic, it is often the codereliction η that is preferred and plays a more central role instead of the deriving transformation d. Therefore, the goal of this paper is to provide an alternative proof that every Lafont category with finite biproducts is a differential category by showing that every free exponential modality comes equipped with a unique codereliction η, which we will construct using the couniversal property of $!A$.

It is always of mathematical interest to have different proofs of the same result, especially when said proofs take different approaches. In this case, the alternative proof presented here has a more differential linear logic "flavour" to it, and should be of use to those who work more with the codereliction rather than the deriving transformation. This will also help clearly unpack the differential structure of an arbitrary Lafont category with finite biproducts, in particular by showing that the differential structure is independent of the construction of the free exponential modality, but depends solely on the couniversal property of the free exponential modality. To prove the desired result, we use the fact that coderelictions are closely linked to !-coalgebras. In [5], Blute, Lucyshyn-Wright, and O'Neill showed that it was possible to construct !-coalgebras using the deriving transformation d. Therefore, it is also possible to construct !-coalgebras using the codereliction η. Readers familiar with the concept of !-coalgebras may think that the codereliction $A \xrightarrow{\eta_A} !A$ is a !-coalgebra structure since it is of the appropriate type. Unfortunately, this is not the case. The reason for this is because, for an arbitrary coalgebra modality, every !-coalgebra is also a cocommutative comonoid. If η_A was a !-coalgebra structure, then A would be a cocommutative comonoid whose comultiplication is given by zero. However, such a comultiplication does not have a counit! To fix this problem, we borrow a trick from Melliès, Tabareau, and Tasson in [27], by considering the free pointed object over A, which in this case is $k \oplus A$, where k is the monoidal unit and $\oplus$ is the biproduct. Then by using the same construction as in [5], we use the codereliction $A \xrightarrow{\eta_A} !A$ to build a !-coalgebra on $k \oplus A$, $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$. A new observation of this paper is that it turns out that the converse is also true!

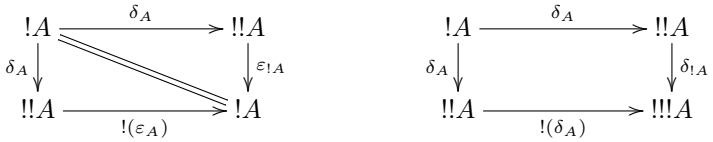
The main new notion of study in this paper is that of an **infinitesimal augmentation**, which is a natural transformation $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ such that H_A is a !-coalgebra and a comonoid morphism. One of the main results of this paper is that there is a bijective correspondence between infinitesimal augmentations and coderelictions (and deriving transformations). A possible advantage of infinitesimal augmentations compared to deriving transformations and coderelictions, is that the notions of !-coalgebras and comonoid morphisms are well-known, even to those who are not familiar with differential categories, and provide yet another way of understanding differentiation via these commonly understood concepts. In fact, it turns out that infinitesimal augmentations are closely linked to the notion of tangent categories [9, 10]. Furthermore, for a free exponential modality, its infinitesimal augmentation is easily constructed, unique, and satisfies the necessary axioms almost automatically simply by construction. We hope that this paper will help open the door to revisiting other examples of Lafont categories and studying them from a differential category point of view, such as, for example, the Lafont categories with infinite biproducts studied by Laird in [20].

Conventions. In this paper, we will use diagrammatic order for composition: this means that the composite map fg is the map which first does f then g. All commutative diagrams drawn in this paper are assumed to commute.

2 Coalgebra Modalities

In this background section, we review the notions of comonads and their coalgebras, comonoids, (monoidal) coalgebra modalities and their coalgebras, and the Seely isomorphisms. We take the time to provide these definitions for readers less familiar with category theory, to introduce notation, and in trying to keep this paper as self-contained as possible. For a more in-depth introduction, we refer the reader to the following introductory sources [26, 29].

▶ **Definition 1** ([26, Section 6.8]). *A **comonad** on a category $\mathbb{X}$ is a triple $(!, \delta, \varepsilon)$ consisting of a functor $\mathbb{X} \xrightarrow{!} \mathbb{X}$ and two natural transformations $!A \xrightarrow{\delta_A} !!A$ and $!A \xrightarrow{\varepsilon_A} A$ such that:*

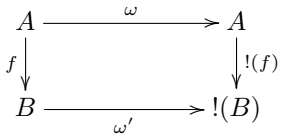


(1)

*A **!-coalgebra** is a pair (A, ω) consisting of an object A and a map $A \xrightarrow{\omega} !A$ such that:*

(2)

*For each object A, the **cofree !-coalgebra** over A is the $!$-coalgebra $(!A, \delta_A)$. A **!-coalgebra morphism** $(A, \omega) \xrightarrow{f} (B, \omega')$ is a map $A \xrightarrow{f} B$ such that:*

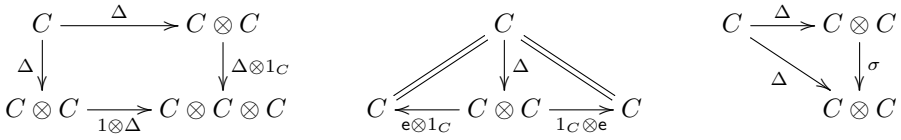


(3)

*The category of $!$-coalgebras and $!$-coalgebra morphisms is denoted $\mathbb{X}^!$ and is also known as the **Eilenberg-Moore category of coalgebras** of the comonad $(!, \delta, \varepsilon)$. There is a forgetful functor $\mathbb{X}^! \xrightarrow{\mathsf{U}^!} \mathbb{X}$, which is defined on objects as $\mathsf{U}^!(A, \omega) = A$ and on maps as $\mathsf{U}^!(f) = f$.*

Coalgebra modalities are comonads on symmetric monoidal categories such that each cofree coalgebra comes equipped with a natural cocommutative comonoid structure. For simplicity, we will work in a symmetric *strict* monoidal category, that is, we will consider the associativity and unit isomorphisms of the monoidal product as strict equalities. For a symmetric monoidal category $\mathbb{X}$, we denote the monoidal product as $\otimes$, the monoidal unit as k, and the natural symmetry isomorphism as $A \otimes B \xrightarrow{\sigma_{A,B}} B \otimes A$. Therefore, $A \otimes k = A = k \otimes A$ and $(A \otimes B) \otimes C = A \otimes B \otimes C = A \otimes (B \otimes C)$.

▶ **Definition 2** ([26, Section 6.3]). *In a symmetric monoidal category $\mathbb{X}$, a **cocommutative comonoid** is a triple (C, Δ, e) consisting of an object C, a map $C \xrightarrow{\Delta} C \otimes C$ called the **comultiplication**, and a map $C \xrightarrow{\mathsf{e}} k$ called the **counit** such that:*



(4)

A ***comonoid morphism*** $(C, \Delta, \mathsf{e}) \xrightarrow{f} (D, \Delta', \mathsf{e}')$ *is a map* $C \xrightarrow{f} D$ *which preserves the comultiplication and counit, that is, the following diagrams commute:*

$$\begin{array}{ccc} C & \xrightarrow{\Delta} & C \otimes C \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f \otimes f} \\ D & \xrightarrow[\Delta']{} & D \otimes D \end{array} \qquad \begin{array}{ccc} C & \xrightarrow{f} & D \\ & {\scriptstyle \mathsf{e}}\searrow & \downarrow{\scriptstyle \mathsf{e}'} \\ & & k \end{array} \tag{5}$$

The category of cocommutative comonoids and comonoid morphisms is denoted $\mathsf{CCom}[\mathbb{X}]$.

For a symmetric monoidal category $\mathbb{X}$, $\mathsf{CCom}[\mathbb{X}]$ is a symmetric monoidal category where the tensor product of cocommutative comonoids is defined as: $(C, \Delta, \mathsf{e}) \otimes (D, \Delta', \mathsf{e}') := \Big(C \otimes D, C \otimes D \xrightarrow{\Delta \otimes \Delta'} C \otimes C \otimes D \otimes D \xrightarrow{1_C \otimes \sigma_{C,D} \otimes 1_D} C \otimes D \otimes C \otimes D, C \otimes D \xrightarrow{\mathsf{e} \otimes \mathsf{e}'} k\Big)$, and the monoidal unit k admits an obvious canonical monoidal structure $(k, 1_k, 1_k)$. In fact, this symmetric monoidal structure on $\mathsf{CCom}[\mathbb{X}]$ is a finite product structure.

▶ **Definition 3** ([3, Definition 1])**.** *A* ***coalgebra modality*** *on a symmetric monoidal category is a quintuple* $(!, \delta, \varepsilon, \Delta, e)$ *consisting of a comonad* $(!, \delta, \varepsilon)$*, a natural transformation* $!A \xrightarrow{\Delta_A} !A \otimes !A$*, and a natural transformation* $!A \xrightarrow{\mathsf{e}_A} K$ *such that for each object* A*, the triple* $(!A, \Delta_A, \mathsf{e}_A)$ *is a cocommutative comonoid and* $(!A, \Delta_A, \mathsf{e}_A) \xrightarrow{\delta_A} (!!A, \Delta_{!A}, \mathsf{e}_{!A})$ *is a comonoid morphism.*

Note that naturality of Δ and e is equivalent to asking that for every map $A \xrightarrow{f} B$, $(!A, \Delta_A, \mathsf{e}_A) \xrightarrow{!(f)} (!B, \Delta_B, \mathsf{e}_B)$ is a comonoid morphism. Furthermore, every !-coalgebra of a coalgebra modality comes equipped with a cocommutative comonoid structure [24, Section 4.1]. Indeed, if (A, ω) is an !-coalgebra, then $(A, \Delta^\omega, \mathsf{e}^\omega)$ is a cocommutative comonoid where the comultiplication and counit are defined as follows:

$$\Delta^\omega := \; A \xrightarrow{\omega} !A \xrightarrow{\Delta_A} !A \otimes !A \xrightarrow{\varepsilon_A \otimes \varepsilon_A} A \otimes A \qquad \mathsf{e}^\omega := \; A \xrightarrow{\omega} !A \xrightarrow{\mathsf{e}_A} k \tag{6}$$

It is important to point out that $(A, \Delta^\omega, \mathsf{e}^\omega)$ is in general only a cocommutative comonoid in the base category $\mathbb{X}$ and not in the coEilenberg-Moore category $\mathbb{X}^!$, since the latter does not necessarily have a monoidal product. Furthermore, since δ_A is a comonoid morphism, when applying this construction to a cofree !-coalgebra $(!A, \delta_A)$ we re-obtain Δ_A and e_A, that is, $\Delta_A^{\delta_A} = \Delta_A$ and $\mathsf{e}_A^{\delta_A} = \mathsf{e}_A$. On top of this, every !-coalgebra morphism becomes a comonoid morphism on the induced comonoid structures, that is, if $(A, \omega) \xrightarrow{f} (B, \omega')$ is a !-coalgebra morphism, then $(A, \Delta^\omega, \mathsf{e}^\omega) \xrightarrow{f} (B, \Delta^{\omega'}, \mathsf{e}^{\omega'})$ is a comonoid morphism. Therefore, this induces a functor from the coEilenberg-Moore category to the category of cocommutative comonoids, $\mathbb{X}^! \xrightarrow{\mathcal{I}^!} \mathsf{CCom}[\mathbb{X}]$. In general, however, $\mathcal{I}^!$ is not equivalence.

We now turn our attention to coalgebra modalities with Seely isomorphisms [2, 4, 30], which requires the symmetric monoidal category to have finite products. For a category with finite products, we denote the binary product of objects by $A \times B$ with projection maps $A \times B \xrightarrow{\pi_0} A$ and $A \times B \xrightarrow{\pi_1} B$, pairing operation $\langle -, - \rangle$, and we denote the chosen terminal object as $\top$, with the unique maps to terminal object as $A \xrightarrow{\mathsf{t}_A} \top$.

▶ **Definition 4** ([3, Definition 10])**.** *In a symmetric monoidal category* $\mathbb{X}$ *with finite products* $\times$ *and terminal object* $\top$*, a coalgebra modality* $(!, \delta, \varepsilon, \Delta, e)$ *has* ***Seely isomorphisms*** *if the natural transformation* $\chi_{A,B} := !(A \times B) \xrightarrow{\Delta_{A \times B}} !(A \times B) \otimes !(A \times B) \xrightarrow{!(\pi_0) \otimes !(\pi_1)} !A \otimes !B$ *and the map* $\chi_\top : !\top \xrightarrow{\mathsf{e}_\top} K$ *are isomorphisms, so* $!\top \cong k$ *and* $!(A \times B) \cong !A \otimes !B$*. A coalgebra*

modality with Seely isomorphisms is called a ***storage modality****. A* ***monoidal storage category*** *(also sometimes known as a (new) Seely category) is a symmetric monoidal category with finite products and a coalgebra modality which has Seely isomorphisms.*

Storage modalities can equivalently be defined as **monoidal coalgebra modalities** [3, Definition 2], which are coalgebra modalities equipped with a natural transformation $!A \otimes !B \xrightarrow{\mathsf{m}_{A,B}} !(A \otimes B)$ and a map $k \xrightarrow{\mathsf{m}_k} !k$ such that the underlying comonad is $!$ a symmetric monoidal comonad [24, Definition 3.8], and that both Δ and e are both monoidal transformations and !-coalgebra morphisms (which imply that $\mathsf{m}_{A,B}$ and m_k are comonoid morphisms). As explained in [2, 4], every storage modality is a monoidal coalgebra modality, where $\mathsf{m}_{A,B} := !A \otimes !B \xrightarrow{\chi^{-1}_{A,B}} !(A \times B) \xrightarrow{\delta_{A \times B}} !!(A \times B) \xrightarrow{!(\chi_{A,B})} !(!A \otimes !B) \xrightarrow{!(\varepsilon_A \otimes \varepsilon_B)} !(A \otimes B)$ and $\mathsf{m}_k := k \xrightarrow{\chi^{-1}_{\top}} !\top \xrightarrow{\delta_{\top}} !!\top \xrightarrow{!(\chi_{\top})} !k$. It is worth mentioning that there are multiple equivalent ways of defining a monoidal coalgebra modality. One characterization, which is of particular important to this paper, is that the monoidal coalgebra modality coherences are precisely what is required so that the tensor product of the base category becomes a product in the coEilenberg-Moore category. This is explained in detail in [29]. Explicitly, the terminal object is the !-coalgebra (k, m_k), while the product, which we denote $\otimes^!$, is the !-coalgebra defined as $(A, \omega) \otimes^! (B, \omega') := \Big(A \otimes B, A \otimes B \xrightarrow{\omega \otimes \omega'} !(A) \otimes !(B) \xrightarrow{\mathsf{m}_{\otimes}} !(A \otimes B)\Big)$. As such, the forgetful functor $\mathbb{X}^! \xrightarrow{\mathsf{U}^!} \mathbb{X}$ preserves the symmetric monoidal structure strictly but not the product structure. On the other hand, $\mathbb{X}^! \xrightarrow{\mathcal{I}^!} \mathsf{CCom}[\mathbb{X}]$ preserves the finite product structure strictly.

There is no shortage of examples of (monoidal) coalgebra modalities since every categorical model of MELL admits a monoidal coalgebra modality/storage modality. For example, Hyland and Schalk provide a nice list of such examples in [18, Section 2.4].

3 Coderelictions

In this section, we review the notion of coderelictions, as well as briefly reviewing differential categories and additive bialgebra modalities. In particular, we highlight the bijective correspondence between coderelictions and deriving transformations. For more details on differential categories, we refer the reader to [6, 3].

The underlying categorical structure of a differential category is not only a symmetric monoidal category but that of an *additive* symmetric monoidal category. Indeed, two of the basic properties of the derivative from classical differential calculus require addition: the Leibniz rule and the constant rule. Therefore we must first discuss the additive structure, and so we begin this section by recalling additive structure by starting with the notion of an additive category. Here we mean "additive" in the Blute, Cockett, and Seely sense of the term [6], that is, enriched over commutative monoids. In particular, we do not assume negatives nor do we assume biproducts which differs from other definitions of an additive category found in the literature.

▶ **Definition 5** ([3, Definition 3])**.** *An* ***additive symmetric monoidal category*** *is a symmetric monoidal category $\mathbb{X}$ such that each hom-set $\mathbb{X}(A, B)$ is a commutative monoid with addition $\mathbb{X}(A, B) \times \mathbb{X}(A, B) \xrightarrow{+} \mathbb{X}(A, B)$, $(f, g) \mapsto f + g$, and zero map $0 \in \mathbb{X}(A, B)$, such that composition and the tensor product preserves the additive structure, that is, the following equalities hold: $k(f+g)h = kfh + kgh$ and $k0h = 0$, and $k \otimes (f+g) \otimes h = k \otimes f \otimes h + k \otimes g \otimes h$ and $k \otimes 0 \otimes h = 0$. An* ***additive storage category*** *is a monoidal storage category which is also an additive symmetric monoidal category.*

We first note that if an additive symmetric monoidal category has finite products, then said finite product structure is in fact a finite biproduct structure that is distributive. We denote the zero object as 0, and the biproduct as $\oplus$ with injection maps $A \xrightarrow{\iota_0} A \oplus B$ and $B \xrightarrow{\iota_1} A \oplus B$, which satisfy the biproduct coherence identities with the projection maps, that is, $\pi_0\iota_0 + \pi_1\iota_1 = 1_{A\oplus B}$, $\iota_0\pi_0 = 1_A$, $\iota_1\pi_1 = 1_B$, $\iota_0\pi_1 = 0$, and $\iota_1\pi_0 = 0$. The additive symmetric monoidal structure guarantees that we also have the distributivity laws between the monoidal and biproduct structures: $(A \oplus B) \otimes (C \oplus D) \cong (A \otimes C) \oplus (A \otimes D) \oplus (B \otimes C) \oplus (B \otimes D)$ and $A \otimes 0 \cong 0 \cong 0 \otimes A$. It is worth mentioning that every symmetric monoidal category with distributive finite biproducts is an additive symmetric monoidal category, and that conversely, every additive symmetric monoidal category has a finite biproduct completion. With all that said, biproducts are not necessary for the axiomatization of a differential category.

Differential categories were introduced by Blute, Cockett, and Seely in [6] to provide a categorical axiomatization of the basic properties of the differentiation, as well as provide categorical models of Ehrhard and Regnier's differential linear logic [13, 14].

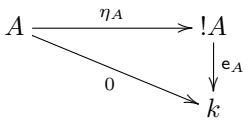
▶ **Definition 6** ([6, Definition 2.4]). *A **differential category** is an additive symmetric monoidal category with a coalgebra modality $(!, \delta, \varepsilon, \Delta, \varepsilon)$ which comes equipped with a **deriving transformation**, that is, a natural transformation $!A \otimes A \xrightarrow{\mathsf{d}_A} !A$ which satisfies the axioms as found in [3, Definition 7]. A **differential storage category** is a differential category with finite products such that its coalgebra modality has Seely isomorphisms.*

The axioms of a deriving transformation include analogues of the product rule, chain rule, that the derivative of a constant map is zero, and that the derivative of a linear map is a constant map. The coKleisli maps of a differential category, that is, the maps of type $!A \xrightarrow{f} B$ are thought of as smooth maps since, in a certain sense, they are differentiable. Indeed, the derivative of a coKleisli map $!A \xrightarrow{f} B$ is the composite $\mathsf{D}[f] := !A \otimes A \xrightarrow{\mathsf{d}_A} !A \xrightarrow{f} B$. This idea is made precise by the fact that the coKleisli category of a differential is a Cartesian differential category [7]. On the other hand, it has been recently shown that the coEilenberg-Moore category of a differential category is a tangent category [9, 10], which we discuss briefly at the end of the next section.

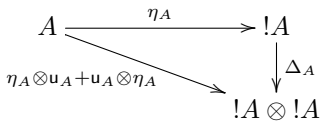
We now turn our attention towards coderelictions. To do so, we must first briefly discuss additive bialgebra modalities. Indeed, for an additive symmetric monoidal category with finite biproducts, a storage modality can equivalently be described as an **additive bialgebra modality** [3, Definition 5], which is briefly a coalgebra modality equipped with natural transformations $!A \otimes !A \xrightarrow{\nabla_A} !A$ and $k \xrightarrow{\mathsf{u}_A} !A$ such that $(!A, \nabla_A, \Delta_A, \mathsf{u}_A, \mathsf{e}_A)$ is a bicommutative bialgebra, and other simple coherences hold. As explained in [3, Section 7], $\nabla_A := !A \otimes !A \xrightarrow{\chi^{-1}_{A,A}} !(A \oplus A) \xrightarrow{!(\pi_0+\pi_1)} !A \otimes !A$ and $\mathsf{u}_A := k \xrightarrow{\chi_0^{-1}} !0 \xrightarrow{!(0)} !A$. For differential storage categories, the differential structure can equivalently be described in terms of a codereliction.

▶ **Definition 7** ([3, Definition 9]). *For an additive storage category with storage coalgebra modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$, a **codereliction** is a natural transformation $A \xrightarrow{\eta_A} !A$ such that:*

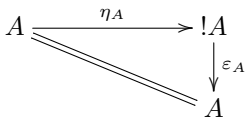
[dC.1] *Constant Rule:*



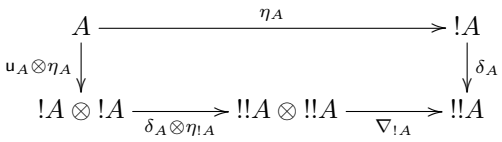
[dC.2] *Product Rule:*



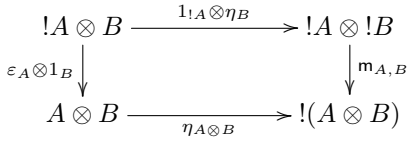
[dC.3] *Linear Rule:*



[**dC.4′**] *Alternative Chain Rule:*



[**dC.m**] *Monoidal Rule:*



We note that the definition of a codereliction provide here is not precisely that found in [3, Definition 9], and is rather defined in terms of the axioms of Fiore's **creation maps** [15]. However, it was shown in [3, Corollary 5] that this axiomatization of a codereliction is equivalent to the original one provided in [6]. As mentioned above, for a storage modality/monoidal coalgebra modality/additive bialgebra modality, coderelictions are in bijective correspondence with deriving transformations.

▶ **Theorem 8** ([3, Theorem 4])**.** *For an additive storage category with storage modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$*, every codereliction induces a deriving transformation, and every deriving transformation induces a codereliction. Explicitly, if* $A \xrightarrow{\eta_A} !A$ *is a codereliction, then* $\mathsf{d}_A := !A \otimes A \xrightarrow{1_{!A} \otimes \eta_A} !A \otimes !A \xrightarrow{\nabla_A} !A$ *is a deriving transformation. Conversely, if* $!A \otimes A \xrightarrow{\mathsf{d}_A} !A$ *is a deriving transformation, then* $\eta_A := A \xrightarrow{\mathsf{u}_A \otimes 1_A} !A \otimes A \xrightarrow{\mathsf{d}_A} !A$ *is a codereliction. Furthermore, these constructions are inverses of each other, and therefore, there is a bijective correspondence between deriving transformations and coderelictions.*

4 Infinitesimal Augmentations

In this section, we introduce the notion of an infinitesimal augmentation, which is the main novel concept of this paper. We will show that infinitesimal augmentations are equivalent to coderelictions, and therefore provide yet another alternative axiomatization for differential categories. At the end of this section, we discuss the terminology behind the name "infinitesimal augmentation" and the relationship to tangent category structure. The majority of proofs for this section can be found in the appendix.

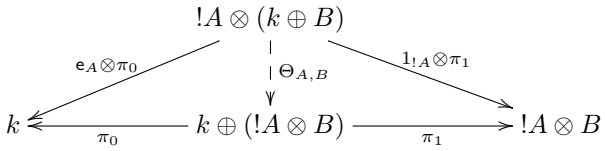
As discussed in the introduction, the basic intuition is that a codereliction $A \xrightarrow{\eta_A} !A$ is not a !-coalgebra structure on A, since A is a missing a comonoid counit. Therefore, we will instead equip $k \oplus A$ with a !-coalgebra. The induced comonoid structure on $k \oplus A$, via the construction of equation (6), should be the canonical one which is conilpotent on the A component. Using element notation, the comultiplication $k \oplus A \xrightarrow{\Lambda_A} (k \oplus A) \otimes (k \oplus A)$ is given by $\Lambda_A(r, a) = (r, 0) \otimes (1, 0) + (0, a) \otimes (1, 0) + (1, 0) \otimes (0, a)$, while the counit is the projection $\pi_0(r, a) = r$. In terms of biproduct distributivity, there are $k \otimes k$, $A \otimes k$ and $k \otimes A$ parts, but no $A \otimes A$ part. So the comultiplication Λ_A is indeed conilpotent on the A component, while the k component is necessary to obtain a counital comonoid.

▶ **Lemma 9.** *In an additive symmetric monoidal category, define the natural transformation* $k \oplus A \xrightarrow{\Lambda_A} (k \oplus A) \otimes (k \oplus A)$ *as* $\Lambda_A := \pi_0(\iota_0 \otimes \iota_0) + \pi_1(\iota_0 \otimes \iota_1) + \pi_1(\iota_1 \otimes \iota_0)$*. Then for every object* A*, the triple* $(k \oplus A, \Lambda_A, \pi_0)$ *is a cocommutative comonoid, and for every map* $A \xrightarrow{f} B$*,* $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{1_k \oplus f} (k \oplus B, \Lambda_B, \pi_0)$ *is a comonoid morphism.*

Proof. This is straightforward to check and we leave it as an exercise for the reader. ◀

In order to define an infinitesimal augmentation, we will require first defining one extra natural transformation. For an additive storage category with storage coalgebra modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$, using the universal property of the product, define the natural transformation

$!A \otimes (k \oplus B) \xrightarrow{\Theta_{A,B}} k \oplus (!A \otimes B)$ as the unique map which makes the following diagram commute:



(7)

Or more simply, using the additive structure: $\Theta_{A,B} = (\mathsf{e}_A \otimes \pi_0)\iota_0 + (1_{!A} \otimes \pi_1)\iota_1$.

▶ **Definition 10.** *For an additive storage category with storage modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$*, an* ***infinitesimal augmentation*** *is a natural transformation* $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ *such that:*
[IA.1] $(k \oplus A, \mathsf{H}_A)$ *is an* !*-coalgebra;*
[IA.2] $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{\mathsf{H}_A} (!(k \oplus A), \Delta_{k\oplus A}, \mathsf{e}_{k\oplus A})$ *is a comonoid morphism;*
[IA.3] $(!A, \delta_A) \otimes^! (k \oplus B, \mathsf{H}_B) \xrightarrow{\Theta_{A,B}} (k \oplus (!A \otimes B), \mathsf{H}_{!A\otimes B})$ *is a* !*-coalgebra morphism.*

The axioms of infinitesimal augmentation are analogous to those of a codereliction. The two diagrams of a !-coalgebra for **[IA.1]** correspond to the Linear Rule **[dC.3]** and the Alternative Chain Rule **[dC.4′]**. The two diagrams of a comonoid morphism for **[IA.2]** correspond to the Constant Rule **[dC.1]** and the Product Rule **[dC.2]**. And lastly, the diagram of a !-coalgebra morphism for **[IA.3]** corresponds to Monoidal Rule **[dC.m]**. It is worth mentioning that it is possible that some of the axioms of an infinitesimal augmentation may be redundant, as was the case for the original definitions of a codereliction and a creation map. However, we've included them here to provide a clear complete story. We now show that the induced comonoid structure from the !-coalgebra structure is the one from Lemma 9.

▶ **Lemma 11.** *If* H *is an infinitesimal augmentation, then for every object* A*:*
(i) $\Delta^{\mathsf{H}_A} = \Lambda_A$ *and* $\mathsf{e}^{\mathsf{H}_A} = \pi_0$*, where* Δ^{H_A} *and* $\mathsf{e}^{\mathsf{H}_A}$ *are defined as in equation (6).*
(ii) $(k, \mathsf{m}_k) \xrightarrow{\iota_0} (k \oplus A, \mathsf{H}_A)$ *and* $(k \oplus A, \mathsf{H}_A) \xrightarrow{\pi_0} (k, \mathsf{m}_k)$ *are* !*-coalgebra morphisms.*

Proof. It follows immediately from **[IA.1]** and **[IA.2]** that (i) holds, which we leave to the reader to check for themselves. For (ii), we use **[IA.1]**, **[IA.2]**, the biproduct identities, naturality of H, and that e is monoidal:

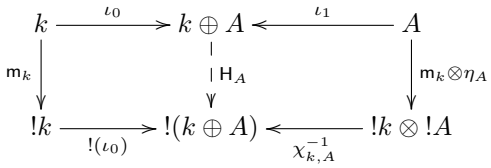
$$\mathsf{H}_A!(\pi_0) = \mathsf{H}_A!(\mathsf{H}_A)!(\mathsf{e}_A) = \mathsf{H}_A\delta_A!(\mathsf{e}_A) = \mathsf{H}_A\mathsf{e}_A\mathsf{m}_k = \pi_0\mathsf{m}_k$$

$$\mathsf{m}_k!(\iota_0) = \iota_0\pi_0\mathsf{m}_k!(\iota_0) = \iota_0\mathsf{H}_A!(\pi_0)!(\iota_0) = \iota_0\mathsf{H}_A!(1_k \oplus 0) = \iota_0(1_k \oplus 0)\mathsf{H}_A = \iota_0\mathsf{H}_A$$

So we conclude that ι_0 and π_0 are !-coalgebra morphisms. ◀

We now show how to construct an infinitesimal augmentation from a codereliction:

▶ **Proposition 12.** *Every codereliction induces an infinitesimal augmentation. Explicitly, for a differential storage category with storage modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ *and codereliction* $A \xrightarrow{\eta_A} !A$*, define the natural transformation* $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ *as the unique map which makes the following diagram commute (using the couniversal property of the coproduct):*



Or equivalently, using the additive structure: $\mathsf{H}_A := \pi_0\mathsf{m}_k!(\iota_0) + \pi_1(\mathsf{m}_k \otimes \eta_A)\chi^{-1}_{k,A}$*. Then* H *is an infinitesimal augmentation.*

Proof. This is actually an application of [5, Theorem 5.1] which we explain in detail in Appendix A. ◀

We now show how to construct a codereliction from an infinitesimal augmentation:

▶ **Proposition 13.** *Every infinitesimal augmentation induces a codereliction. Explicitly, for an additive storage category with coalgebra modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ *which has Seely isomorphisms and infinitesimal augmentation* $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$*, define* $\eta_A := A \xrightarrow{\iota_1} k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A) \xrightarrow{!(\pi_1)} !A$*. Then* η *is a codereliction. Therefore, an additive storage category whose storage modality has an infinitesimal augmentation is a differential storage category.*

Proof. See Appendix B. ◀

We now state the first main result of this paper:

▶ **Theorem 14.** *For the storage modality of additive storage category, there is a bijective correspondence between coderelictions and infinitesimal augmentations.*

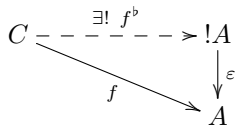
Proof. See Appendix C, where we show that constructions of Proposition 12 and Proposition 13 are inverses of each other. ◀

We conclude this section with a discussion on the terminology behind the name "infinitesimal augmentation". "Augmentation" is a reference to the fact that $k \oplus A$ is always an augmented (co)algebra in the classical sense, in particular since $k \oplus A$ is the (co)free (co)pointed object over A. "Infinitesimal" is related to tangent category terminology. A tangent category [9] is a category equipped with an endofunctor T and various other natural transformations whose axioms generalize the theory of smooth manifolds and their tangent bundles, with the category of smooth manifolds being the canonical example. A representable tangent category is a tangent category with finite products and such that T is a representable functor, that is, $\mathsf{T} \cong (-)^D$ for some exponent object D. The object D is called an **infinitesimal object**. In [10, Section 6] it was shown that, under a mild limit condition, the coEilenberg-Moore category of a differential storage category is a representable tangent category whose infinitesimal object is $(k \oplus k, \mathsf{H}_k)$. Therefore, the !-coalgebra structure of the infinitesimal object is precisely the infinitesimal augmentation. In future work, it would be interesting to further study the connection between infinitesimal augmentations and tangent structure. In particular, infinitesimal augmentations may provide the key in generalizing linear-non-linear adjunctions [2, 25, 26] for differential categories (where one would replace a Cartesian category with a tangent category).

5 Coderelictions for Free Exponential Modalities

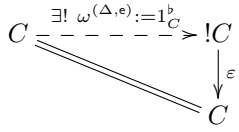
In this section we provide the main objective of this paper, that is, we provide an alternative proof that every additive Lafont category with finite biproducts is a differential storage category. In particular, we will explain how to construct the (necessarily unique) codereliction and induced deriving transformation of the free exponential modality using its couniversal property. In fact, we will first show that every free exponential modality has an infinitesimal augmentation, which is easily constructed using the couniversal property.

▶ **Definition 15** ([27]). *A **free exponential modality** is a coalgebra modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ such that for each object A, $!A$ is a **cofree cocommutative comonoid** over A, that is, if (C, Δ, e) is a comonoid then for every map $C \xrightarrow{f} A$, there exists a unique comonoid morphism $(C, \Delta, \mathsf{e}) \xrightarrow{f^\flat} (!A, \Delta, \mathsf{e})$ such that the following diagram commutes:*



*A **(additive) Lafont category** is a (additive) symmetric monoidal category with a free exponential modality.*

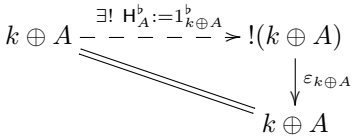
We should note that here we are using the term "Lafont category" in the sense of Blute, Cockett, and Seely as in [4], which is the same as in [25] but which drops the closed structure requirement. The coEilenberg-Moore category of a free exponential modality is isomorphic to the category of cocommutative comonoids. In other words, for a free exponential modality, every cocommutative comonoid is a !-coalgebra. Explicitly, if (C, Δ, e) is a cocommutative comonoid, then define $(C, \Delta, \mathsf{e}) \xrightarrow{\omega^{(\Delta,\mathsf{e})}} (!C, \Delta_C, \mathsf{e}_C)$ as the unique comonoid morphism such that the following diagram commutes:



Then it follows that $(C, \omega^{(\Delta,\mathsf{e})})$ is a !-coalgebra. Furthermore, if $(C, \Delta, \mathsf{e}) \xrightarrow{f} (D, \Delta', \mathsf{e}')$ is a comonoid morphism, then $(C, \omega^{(\Delta,\mathsf{e})}) \xrightarrow{f} (D, \omega^{(\Delta',\mathsf{e}')})$ is a !-coalgebra morphism. On top of this, it follows that $\Delta^{\omega^{(\Delta,\mathsf{e})}} = \Delta$ and $\mathsf{e}^{\omega^{(\Delta,\mathsf{e})}} = \mathsf{e}$, where $\Delta^{\omega^{(\Delta,\mathsf{e})}}$ and $\mathsf{e}^{\omega^{(\Delta,\mathsf{e})}}$ are defined as in (6). Therefore, this induces a functor $\mathcal{J}^! : \mathsf{CCom}[\mathbb{X}] \to \mathbb{X}^!$ which is inverse to $\mathcal{I}^! : \mathbb{X}^! \to \mathsf{CCom}[\mathbb{X}]$. Furthermore, it is a well-known fact that free exponential modalities are always monoidal coalgebra modalities [2, 25]. Explicitly, $(k, 1_k, 1_k) \xrightarrow{\mathsf{m}_k} (!k, \Delta_k, \mathsf{e}_k)$ and $(!A, \Delta_A, \mathsf{e}_A) \otimes (!B, \Delta_B, \mathsf{e}_B) \xrightarrow{\mathsf{m}_{A,B}} (!(A \otimes B), \Delta_{A\otimes B}, \mathsf{e}_{A\otimes B})$ are the unique comonoid morphisms defined respectively as $\mathsf{m}_k := 1_k^\flat$ and $\mathsf{m}_{A,B} := (\varepsilon_A \otimes \varepsilon_B)^\flat$. As such, in the presence of finite products, every free exponential modality has Seely isomorphisms and is therefore a storage modality. Explicitly, $(!A, \Delta_A, \mathsf{e}_A) \otimes (!B, \Delta_B, \mathsf{e}_B) \xrightarrow{\chi^{-1}_{A,B}} (!(A \times B), \Delta_{A\times B}, \mathsf{e}_{A\times B})$ and $(k, 1_k, 1_k) \xrightarrow{\chi^{-1}_\top} (!\top, \Delta_\top, \mathsf{e}_\top)$ are the unique comonoid morphisms defined respectively as $\chi^{-1}_{A,B} := \langle \varepsilon_A \otimes \mathsf{e}_B, \mathsf{e}_A \otimes \varepsilon_B \rangle^\flat$ and $\mathsf{t}_k^\flat$. For an additive Lafont category, it follows that every free exponential modality is thus also an additive bialgebra modality. Explicitly, $(!A, \Delta_A, \mathsf{e}_A) \otimes (!A, \Delta_A, \mathsf{e}_A) \xrightarrow{\nabla_A} (!A, \Delta_A, \mathsf{e}_A)$ and $(k, 1_k, 1_k) \xrightarrow{\mathsf{u}_A} (!A, \Delta_A, \mathsf{e}_A)$ are the unique comonoid morphisms defiend respectively as $\nabla_A := (\varepsilon_A \otimes \mathsf{e}_A + \mathsf{e}_A \otimes \varepsilon_A)^\flat$ and $\mathsf{u}_A := 0^\flat$. So in particular, in the presence of additive structure, for a free exponential modality, $!A$ is also a bicommutative bialgebra.

We now turn our attention to constructing the infinitesimal augmentation for the free exponential modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ of an additive Lafont category with finite biproducts. As shown in Lemma 9, $(k \oplus A, \Lambda_A, \pi_0)$ is a cocommutative comonoid, and therefore admits a canonical !-coalgebra structure. Define the natural transformation $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ as $\mathsf{H}_A := \omega^{(\Lambda,\pi_0)}$, that is, $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{\mathsf{H}_A} (!(k \oplus A), \Delta_{k\oplus A}, \mathsf{e}_{k\oplus A})$ is the unique comonoid

morphism such that the following diagram commutes:



(8)

We now carefully show in steps that H is indeed an infinitesimal augmentation. Starting with the an important, but often overlooked step, of showing that H is natural.

▶ **Lemma 16.** H *is a natural transformation.*

Proof. Consider a map $A \xrightarrow{f} B$. By Lemma 9, $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{1_k \oplus f} (k \oplus B, \Lambda_B, \pi_0)$ is a comonoid morphism. Therefore, $(1_k \oplus f)\mathsf{H}_B$ and $\mathsf{H}_A!(1_k \oplus f)$ are comonoid morphisms of the same type $(k \oplus A, \Lambda_A, \pi_0) \to (!(k \oplus A), \Delta_{k\oplus A}, \mathsf{e}_{k\oplus A})$. However, we easily compute that:

$$\mathsf{H}_A!(1_k \oplus f)\varepsilon_{k\oplus B} = \mathsf{H}_A\varepsilon_A(1_k \oplus f) = (1_k \oplus f) = (1_k \oplus f)\mathsf{H}_B\varepsilon_{k\oplus B}$$

Since $\mathsf{H}_A!(1_k \oplus f)\varepsilon_{k\oplus B} = (1_k \oplus f)\mathsf{H}_B\varepsilon_{k\oplus B}$, it follows from the couniversal property of $!(k \oplus B)$ that $\mathsf{H}_A!(1_k \oplus f) = (1_k \oplus f)\mathsf{H}_B$. So we conclude that H is a natural transformation. ◀

Next, it follows that **[IA.1]** and **[IA.2]** are automatic by construction.

▶ **Lemma 17.** *For every object A,*

(i) $(k \oplus A, \mathsf{H}_A)$ *is an !-coalgebra;*

(ii) $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{\mathsf{H}_A} (!(k \oplus A), \Delta_{k\oplus A}, \mathsf{e}_{k\oplus A})$ *is a comonoid morphism.*

Proof. Both are automatic by construction since $\mathsf{H}_A := \omega^{(\Lambda,\pi_0)}$. ◀

For the free exponential modality, !-coalgebra morphisms correspond to comonoid morphism. Therefore, in order to prove **[IA.3]**, it is sufficient to show that $\Theta_{A,B}$ is a comonoid morphism of the appropriate type. To do so, we will require the following lemma:

▶ **Lemma 18.** *Let (C, Δ, e) be a cocommutative comonoid. Then for every object A, $(C, \Delta, \mathsf{e}) \xrightarrow{F} (k \oplus A, \Lambda_A, \pi_0)$ is a comonoid morphism if and only if $C \xrightarrow{F} k \oplus A$ is of the form $F = \langle \mathsf{e}, f \rangle$ for some map $C \xrightarrow{f} A$ such that $\Delta(f \otimes f) = 0$.*

Proof. Recall that any map $C \xrightarrow{F} k \oplus A$ satisfies $F = \langle F\pi_0, F\pi_1 \rangle$. Then suppose that $(C, \Delta, \mathsf{e}) \xrightarrow{F} (k \oplus A, \Lambda_A, \pi_0)$ is a comonoid morphism. Since F preserves the counit, it follows that $F\pi_0 = \mathsf{e}$. Next, note that by definition $\Lambda_A(\pi_1 \otimes \pi_1) = 0$. Then since F preserves the comultiplication it follows that $\Delta(F \otimes F)(\pi_1 \otimes \pi_1) = F\Lambda_A(\pi_1 \otimes \pi_1) = 0$. Therefore, $F\pi_1$ satisfies the desired equality, and so $F = \langle \mathsf{e}, F\pi_1 \rangle$ is of the desired form. Conversely, suppose that f is a map which satisfies $\Delta(f \otimes f) = 0$. By definition, it is automatic that $\langle \mathsf{e}, f \rangle$ preserves the counit since $\langle \mathsf{e}, f \rangle\pi_0 = \mathsf{e}$. Next we need to show that $\langle \mathsf{e}, f \rangle$ also preserves the comultiplication. To do so, we will show that $\langle \mathsf{e}, f \rangle\Lambda_A(\pi_i \otimes \pi_j) = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)(\pi_i \otimes \pi_j)$ for $i, j \in \{0, 1\}$.

$$\langle \mathsf{e}, f \rangle\Lambda_A(\pi_0 \otimes \pi_0) = \langle \mathsf{e}, f \rangle\pi_0 = \mathsf{e} = \Delta(\mathsf{e} \otimes \mathsf{e}) = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)(\pi_0 \otimes \pi_0)$$

$$\langle \mathsf{e}, f \rangle\Lambda_A(\pi_0 \otimes \pi_1) = \langle \mathsf{e}, f \rangle\pi_1 = f = \Delta(\mathsf{e} \otimes f) = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)(\pi_0 \otimes \pi_1)$$

$$\langle \mathsf{e}, f \rangle\Lambda_A(\pi_1 \otimes \pi_0) = \langle \mathsf{e}, f \rangle\pi_1 = f = \Delta(f \otimes \mathsf{e}) = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)(\pi_1 \otimes \pi_0)$$

$$\langle \mathsf{e}, f \rangle\Lambda_A(\pi_1 \otimes \pi_1) = 0 = \Delta(f \otimes f) = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)(\pi_1 \otimes \pi_1)$$

Then by the distributivity of the biproduct and the universal property of the product, it follows that $\langle \mathsf{e}, f \rangle\Lambda_A = \Delta\left(\langle \mathsf{e}, f \rangle \otimes \langle \mathsf{e}, f \rangle\right)$. Therefore, $\langle \mathsf{e}, f \rangle$ is a comonoid morphism. ◀

▶ **Corollary 19.** *$(!A, \Delta_A, \mathsf{e}_A) \otimes (k \oplus B, \Lambda_B, \pi_0) \xrightarrow{\Theta_{A,B}} (k \oplus (!A \otimes B), \Lambda_{!A \otimes B}, \pi_0)$ is a comonoid morphism where $!A \otimes (k \oplus B) \xrightarrow{\Theta_{A,B}} k \oplus (!A \otimes B)$ is defined as in (7). Therefore, we also have that $(!A, \delta_A) \otimes^{!} (k \oplus B, \mathsf{H}_B) \xrightarrow{\Theta_{A,B}} (k \oplus (!A \otimes B), \mathsf{H}_{!A \otimes B})$ is a !-coalgebra morphism.*

Proof. By construction $\Theta_{A,B} = \langle \mathsf{e}_A \otimes \pi_0, 1_{!A} \otimes \pi_1 \rangle$. Then by Lemma 18, to show that $\Theta_{A,B}$ is a comonoid morphism, it suffices to show that $1_{!A} \otimes \pi_1$ satisfies the extra identity, since the first component of $\Theta_{A,B}$ is indeed the counit of $!A \otimes (k \oplus B)$. However by naturality of the symmetry isomorphism and that $\Lambda_B(\pi_1 \otimes \pi_1) = 0$, we easily see that (we omit the subscripts for space):

$$(\Delta \otimes \Lambda)(1 \otimes \sigma \otimes 1)(1 \otimes \pi_1 \otimes 1 \otimes \pi_1) = (\Delta \otimes \Lambda)(1 \otimes 1 \otimes \pi_1 \otimes \pi_1)(1 \otimes \sigma \otimes 1) = 0$$

So $\Theta_{A,B}$ is a comonoid morphism. Furthermore, it is straightforward to check that for $(!A, \Delta_A, \mathsf{e}_A) \otimes (k \oplus B, \Lambda_B, \pi_0)$, its associated !-coalgebra is precisely $(!A, \delta_A) \otimes^{!} (k \oplus B, \mathsf{H}_B)$. Therefore, since every comonoid morphism is also a !-coalgebra morphism between the induced !-coalgebras, it follows that $\Theta_{A,B}$ is a !-coalgebra morphism. ◀

Bringing all of the above lemmas and corollary together, we obtain:

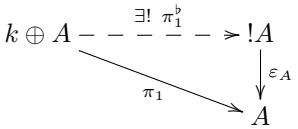
▶ **Proposition 20.** *For an additive Lafont category with free exponential modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ and finite biproducts, H as defined in equation (8) is an infinitesimal augmentation, and furthermore it is the unique infinitesimal augmentation for the free exponential modality.*

Proof. Lemma 16 shows that H is a natural transformation, while Lemma 17 and Corollary 19 show that H satisfies **[IA.1]** and **[IA.2]** respectively. So H is indeed an infinitesimal augmentation. Now suppose that H' was another infinitesimal augmentation. By Lemma 11, we have that $\Delta^{\mathsf{H}_A} = \Lambda_A = \Delta^{\mathsf{H}'_A}$ and $\mathsf{e}^{\mathsf{H}_A} = \pi_0 = \mathsf{e}^{\mathsf{H}'_A}$. Therefore, the !-coalgebras $(k \oplus A, \mathsf{H}_A)$ and $(k \oplus A, \mathsf{H}'_A)$ both induce the same cocommutative comonoid $(k \oplus A, \Lambda_A, \pi_0)$. However, since the coEilenberg-Moore category of the free exponential modality is isomorphic to the category of cocommutative comonoids, it follows that $(k \oplus A, \mathsf{H}_A) = (k \oplus A, \mathsf{H}'_A)$. Therefore, $\mathsf{H}_A = \mathsf{H}'_A$. So we conclude that H is the unique infinitesimal augmentation for the free exponential modality. ◀

Therefore, we obtain an alternative proof of Blute, Lucyshyn-Wright, and O'Neill in terms of coderelictions and differential categories (rather than deriving transformations and codifferential categories) which is the main contribution of this paper.

▶ **Theorem 21** ([5, Theorem 4.4])**.** *For an additive Lafont category with free exponential modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ and finite biproducts, the free exponential modality comes equipped with a unique coderelictions $A \xrightarrow{\eta_A} !A$ defined as follows, where $(k \oplus A, \Lambda_A, \pi_0) \xrightarrow{\pi_1^\flat} (!A, \Delta_A, \mathsf{e}_A)$ is the unique comonoid morphism such that the diagram on the right commutes:*

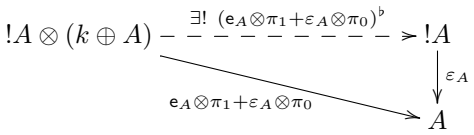
$$\eta_A = \; A \xrightarrow{\iota_1} k \oplus A \xrightarrow{\pi_1^\flat} !A$$



Furthermore, this coderelictions is precisely the induced coderelictions from the infinitesimal augmentation H from Proposition 20 via the construction of Proposition 13. The (necessarily unique) deriving transformation $!A \otimes A \xrightarrow{\mathsf{d}_A} !A$ induced by the construction of Theorem 8 is equal to the following composition,

$$\mathsf{d}_A = \; !A \otimes A \xrightarrow{1_{!A} \otimes \iota_1} !A \otimes (k \oplus A) \xrightarrow{(\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat} !A$$

where $(\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat : (!A, \Delta_A, \mathsf{e}_A) \otimes (k \oplus A, \Lambda_A, \pi_0) \to (!A, \Delta_A, \mathsf{e}_A)$ *is the unique comonoid morphism such that the following diagram commutes:*



Therefore, every additive Lafont category with finite biproducts is a differential category.

Proof. By Proposition 20, H is an infinitesimal augmentation and so, by Proposition 13, H induces a coderelictions η defined as $\eta_A = \iota_1 \mathsf{H}_A!(\pi_1)$. Since H is the unique infinitesimal augmentation, by the bijective correspondence of Theorem 14, it follows that η must also be the unique coderelictions for the free exponential modality. Next, in order to show the desired equality, it suffices to show that $\pi_1^\flat = \mathsf{H}_A!(\pi_1)$. To do so, first note that by definition, $\pi^\flat$ and $\mathsf{H}_A!(\pi_1)$ are comonoid morphisms of the same type $(k \oplus A, \Lambda_A, \pi_0) \to (!A, \Delta_A, \mathsf{e}_A)$. Also note that we have the following equality:

$$\mathsf{H}_A!(\pi_1)\varepsilon_A = \mathsf{H}_A\varepsilon_A\pi_1 = \pi_1$$

Since $\mathsf{H}_A!(\pi_1)\varepsilon_A = \pi_1 = \pi_1^\flat\varepsilon_A$, by the couniversal property of $!A$, it follows that $\mathsf{H}_A!(\pi_1) = \pi_1^\flat$. Therefore, we have that:

$$\eta_A = \iota_1 \mathsf{H}_A!(\pi_1) = \iota_1\pi_1^\flat$$

Thus we conclude that $\eta_A = \iota_1\pi_1^\flat$. By Theorem 8, the coderelictions η induces a deriving transformation d defined as $\mathsf{d}_A = (1_{!A} \otimes \eta_A)\nabla_A$. In order to show to the desired equality, it suffices to show that $(1_{!A} \otimes \pi_1^\flat)\nabla_A = (\mathsf{e}_A \otimes \pi_1)^\flat$. Using again the same strategy as before, we note that $(1_{!A} \otimes \pi_1^\flat)\nabla_A$ and $(\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat$ are comonoid morphisms of the same type $(!A, \Delta_A, \mathsf{e}_A) \otimes (k \oplus A, \Lambda_A, \pi_0) \to (!A, \Delta_A, \mathsf{e}_A)$. We also compute the following equality:

$$(1_{!A} \otimes \pi_1^\flat)\nabla_A\varepsilon_A = (1_{!A} \otimes \pi_1^\flat)(\mathsf{e}_A \otimes \varepsilon_A) + (1_{!A} \otimes \pi_1^\flat)(\varepsilon_A \otimes \mathsf{e}_A) = \mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0$$

Since $(1_{!A} \otimes \pi_1^\flat)\nabla_A\varepsilon_A = \mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0 = (\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat\varepsilon_A$, by the couniversal property of $!A$, it follows that $(1_{!A} \otimes \pi_1^\flat)\nabla_A = (\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat$. Therefore, we have that:

$$\mathsf{d}_A = (1_{!A} \otimes \eta_A)\nabla_A = (1_{!A} \otimes \iota_1)(1 \otimes \pi_1^\flat)\nabla_A = (1_{!A} \otimes \iota_1)(\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat$$

Thus we conclude that $\mathsf{d}_A = (1_{!A} \otimes \iota_1)(\mathsf{e}_A \otimes \pi_1 + \varepsilon_A \otimes \pi_0)^\flat$. ◀

6 Examples

In this section, we provide some examples of free exponential modalities and their coderelictions. Other interesting examples of differential categories with free exponential modalities are studied in [3, 5, 6, 8, 12].

▶ **Example 22.** Let REL be the category of sets and relations, where recall that the tensor product is given by the Cartesian product of sets, $X \otimes Y = X \times Y$, the monoidal unit is a chosen singleton, $k = \{*\}$, the biproduct is given by the disjoint union of sets, $X \oplus Y = X \sqcup Y$. REL is also a Lafont category where for a set X, $!X$ is the set of finite multisets of elements of X. We will denote finite multisets as $[\![x_1, \ldots, x_n]\!]$, $x_i \in X$, where recall that we can have multiple copies of the same element in a finite multiset. The coderelictions is the relation which

associates an element of X to the bag containing only said element, $\eta_X := \{(x, [\![x]\!]) |\ \forall x \in X\} \subseteq X \times !X$. The deriving transformation is the relation which adds an element into the bag, $\mathsf{d}_X := \{(([\![x_1, \ldots, x_n]\!], x), [\![x_1, \ldots, x_n, x]\!]) \mid \forall x_i, x \in X\}$. The infinitesimal extension relates the element of the singleton to all possible bags of copies of the singleton element, and relates an element of X to bags of copies of the singleton element with said element added in: $\mathsf{H}_X := \{(*, [\![\underbrace{*, \ldots, *}_{n\text{-copies}}]\!]) |\ \forall n \in \mathbb{N}\} \cup \{(x, [\![\underbrace{*, \ldots, *}_{n\text{-copies}}, x]\!]) |\ \forall x \in X, n \in \mathbb{N}\} \subseteq (\{*\} \sqcup X) \times !(\{*\} \sqcup X)$. For more details on this example, see [6, Section 2.5.1].

▶ **Example 23.** Let k be a field, and let VEC_k be the category of k-vector spaces and k-linear maps between them. Its dual VEC_k^{op} is a Lafont category where for a k-vector space V, $!V = \mathsf{Sym}(V)$, the free symmetric algebra over V. Note that in VEC_k, $\mathsf{Sym}(V)$ is the free commutative k-algebra over V, and therefore $\mathsf{Sym}(V)$ is the cofree cocommutative comonoid over V in VEC_k^{op}. In particular, if X is a basis of V, then $\mathsf{Sym}(V) \cong k[X]$, where the latter is the polynomial ring over X. We will express η, d, and H in terms of polynomials, and if their types look backwards, it is because we expressing them in VEC_k. The codereliction $K[X] \xrightarrow{\eta_V} V$ is defined as picking out the degree 1 terms of the polynomial, that is, its x_i terms. This can be described as follows: $\eta_V(p(\vec{x})) = \sum_{i=1}^{n} \frac{\partial p(\vec{x})}{\partial x_i}(\vec{0}) x_i$. Note that evaluating a polynomial at zero extracts its constant term. The constant term of $\frac{\partial p(\vec{x})}{\partial x_i}$ is precisely the scalar factor of x_i. Therefore, $\frac{\partial p(\vec{x})}{\partial x_i}(0) x_i$ are precisely the degree 1 terms of $p(\vec{x})$. The deriving transformation $k[X] \xrightarrow{\mathsf{d}_V} k[X] \otimes V$ is defined as mapping a polynomial to its sum of its partial derivatives: $\mathsf{d}_V(p(\vec{x})) = \sum_{i=1}^{n} \frac{\partial p(\vec{x})}{\partial x_i} \otimes x_i$. For the infinitesimal extension, note that $\mathsf{Sym}(k \oplus V) \cong k[X, y]$, therefore $k[X, y] \xrightarrow{\mathsf{H}_V} k \oplus V$ is defined as $\mathsf{H}_V(p(\vec{x}, y)) = p(\vec{0}, 1) + \sum_{i=1}^{n} \frac{\partial p(\vec{x}, y)}{\partial x_i}(\vec{0}, 1) x_i$. We note that this example can be generalized to the category of modules over any commutative semiring. For more details on this example, see [6, Section 2.5.3].

▶ **Example 24.** Example 22 and Example 23 are in fact examples of the same general construction of a free exponential modality given by the product of the symmetric tensor powers, that is, $!A = \prod_{n \in \mathbb{N}} \mathsf{S}_n(A)$, where $\prod_{n \in \mathbb{N}}$ is the countable product and $\mathsf{S}_n(A)$ is the equalizer of all permutations $A^{\otimes^n} \xrightarrow{\cong} A^{\otimes^n}$. The codereliction is defined as the "injection" of A into $!A$ since $\mathsf{S}_1(A) = A$, that is, let $!A \xrightarrow{\pi_n} \mathsf{S}_n(A)$ be the projection map of the product (where note that $!A \xrightarrow{\pi_1} A$), then the codereliction $A \xrightarrow{\eta_A} !(A)$ is defined as the unique map (using the universal property of the product) such that $\eta\pi_1 = 1_A$ and $\eta\pi_n = 0$ for $n \neq 1$. We stress that not all free exponential modalities arise in this manner, as explained in [27].

▶ **Example 25.** Let k be a field, then VEC_k is a Lafont category where for a k-vector space V, $!V$ is the cofree cocommutative k-coalgebra over V. When k is algebraically closed and has characteristic zero (such as $\mathbb{C}$), then $!V$ admits a nice expression [28]: if X is a basis for V, then $!V \cong \bigoplus_{v \in V} k[X]$. In this case, the codereliction $V \xrightarrow{\eta_V} \bigoplus_{v \in V} k[X]$ maps basis elements $x \in X$ to the monomial in the $0 \in V$ component: $\eta_V(x) = (x)_0$. This differential category, and its resulting model of differential linear logic, was studied in detail by Clift and Murfet in [8]. We note that this example can be generalized to the category of modules over any commutative semiring, though the cofree cocommutative coalgebra may not have as nice a form. It is also important to observe that this example is not of the form of Example 24.

References

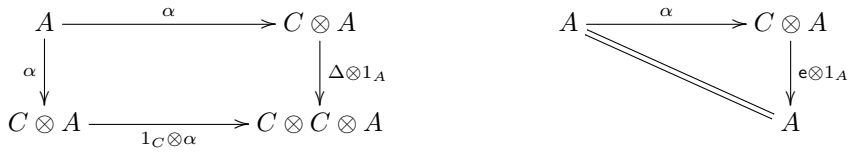
1 M. Barr. Coalgebras over a commutative ring. *Journal of Algebra*, 32(3):600–610, 1974.
2 G.M. Bierman. What is a categorical model of intuitionistic linear logic? In *International Conference on Typed Lambda Calculi and Applications*, pages 78–93. Springer, 1995.
3 R. F. Blute, J. R. B. Cockett, J.-S. P. Lemay, and R. A. G. Seely. Differential categories revisited. *Applied Categorical Structures*, 28:171–235, 2020. doi:10.1007/s10485-019-09572-y.
4 R. F. Blute, J. R. B. Cockett, and R. A. G. Seely. Cartesian differential storage categories. *Theory and Applications of Categories*, 30(18):620–686, 2015.
5 R. F Blute, R. B. B. Lucyshyn-Wright, and K O'Neill. Derivations in codifferential categories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 57:243–280, 2016.
6 R. F. Blute, Blute, J. R. B. Cockett, and R. A. G. Seely. Differential categories. *Mathematical structures in computer science*, 16(6):1049–1083, 2006.
7 R. F. Blute, Blute, J. R. B. Cockett, and R. A. G. Seely. Cartesian differential categories. *Theory and Applications of Categories*, 22(23):622–672, 2009.
8 J. Clift and D. Murfet. Cofree coalgebras and differential linear logic. *Mathematical Structures in Computer Science*, 30(4):416–457, 2020. doi:10.1017/S0960129520000134.
9 J. R. B. Cockett and G. S. H. Cruttwell. Differential structure, tangent structure, and SDG. *Applied Categorical Structures*, 22(2):331–417, 2014.
10 J.R. B. Cockett, J.-S. P. Lemay, and R. B. B. Lucyshyn-Wright. Tangent Categories from the Coalgebras of Differential Categories. In *28th EACSL Annual Conference on Computer Science Logic (CSL 2020)*, volume 152 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2020.17.
11 R. Crubillé, T. Ehrhard, M. Pagani, and C. Tasson. The free exponential modality of probabilistic coherence spaces. In *International Conference on Foundations of Software Science and Computation Structures*, pages 20–35. Springer, 2017.
12 T. Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, 28:995–1060, 2018.
13 T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.
14 T. Ehrhard and L. Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006.
15 M.P. Fiore. Differential structure in models of multiplicative biadditive intuitionistic linear logic. In *International Conference on Typed Lambda Calculi and Applications*, pages 163–177. Springer, 2007.
16 J.-Y. Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.
17 J.-Y. Girard and Y. Lafont. Linear logic and lazy computation. In *International Joint Conference on Theory and Practice of Software Development*, pages 52–66. Springer, 1987.
18 M. Hyland and A. Schalk. Glueing and orthogonality for models of linear logic. *Theoretical computer science*, 294(1-2):183–231, 2003.
19 Y. Lafont. *Logiques, catégories et machines*. PhD thesis, PhD thesis, Université Paris 7, 1988.
20 J. Laird. Weighted models for higher-order computation. *Information and Computation*, 275:104645, 2020.
21 J. Laird, G. Manzonetto, and G. McCusker. Constructing differential categories and deconstructing categories of games. *Information and Computation*, 222:247–264, 2013.
22 J. Laird, G. Manzonetto, G. McCusker, and M. Pagani. Weighted relational models of typed lambda-calculi. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 301–310. IEEE Computer Society, 2013.
23 S. Lang. Algebra, revised 3rd ed. *Graduate Texts in Mathematics*, 211, 2002.
24 J.-S. P. Lemay. Lifting Coalgebra Modalities and MELL Model Structure to Eilenberg-Moore Categories. *Logical Methods in Computer Science*, Volume 15, Issue 4, November 2019. doi:10.23638/LMCS-15(4:8)2019.

25 P.-A. Melliès. Categorical models of linear logic revisited. *HAL preprint, hal-00154229*, 2003.
26 P.-A. Melliès. Categorical semantics of linear logic. *Panoramas et syntheses*, 27:15–215, 2009.
27 P.-A. Melliès, N. Tabareau, and C. Tasson. An explicit formula for the free exponential modality of linear logic. *Mathematical Structures in Computer Science*, pages 1–34, 2017.
28 D. Murfet. On Sweedler's cofree cocommutative coalgebra. *Journal of Pure and Applied Algebra*, 219(12):5289–5304, 2015.
29 A. Schalk. What is a categorical model of linear logic? *Manuscript, available from http://www.cs.man.ac.uk/ schalk/notes/llmodel.pdf*, 2004.
30 R. A. G. Seely. Linear logic, *-autonomous categories and cofree coalgebras. In *Categories in Computer Science and Logic (1987)*, pages 371–382. American Mathematical Society, 1989.
31 S. Slavnov. On banach spaces of sequences and free linear logic exponential modality. *Mathematical Structures in Computer Science*, 29(2):215–242, 2019. doi:10.1017/S0960129517000251.
32 M. E. Sweedler. Hopf algebras. mathematical lecture note series, 1969.

A Proof of Proposition 12

The key to this proof is that we make use of the dual of [5, Theorem 5.1]. To do so, we must first recall the definition of comodules of a cocommutative comonoid.

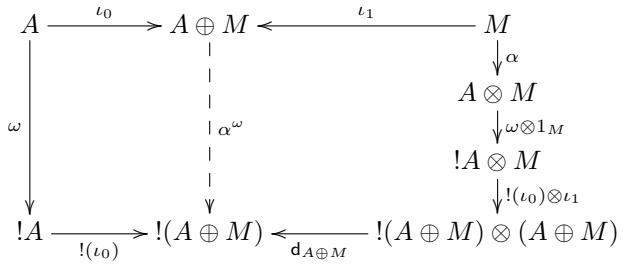
▶ **Definition 26.** *In a symmetric monoidal category, for a cocommutative comonoid (C, Δ, e), a (C, Δ, e)-**comodule** is a pair (M, α) consisting of an object M and a map $M \xrightarrow{\alpha} C \otimes M$ called the **coaction**, such that:*



*For a coalgebra modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ and a $!$-coalgebra (A, ω), a (A, ω)-**comodule** is a $(A, \Delta^\omega, \mathsf{e}^\omega)$-comodule, where Δ^ω and e^ω are defined as in (6).*

As explained in [5], one can use the deriving transformation d to construct new $!$-coalgebras using $!$-coalgebras and their comodules.

▶ **Theorem 27** ([5, Theorem 5.1, Proposition 5.4])**.** *In a differential category with coalgebra modality $(!, \delta, \varepsilon, \Delta, \mathsf{e})$, deriving transformation d, and finite biproducts $\oplus$, if (A, ω) is a $!$-coalgebra and (M, α) a (A, ω)-comodule, define the map $A \oplus M \xrightarrow{\alpha^\omega} !(A \oplus M)$ as the unique map which makes the following diagram commute (using the couniversal property of the coproduct):*



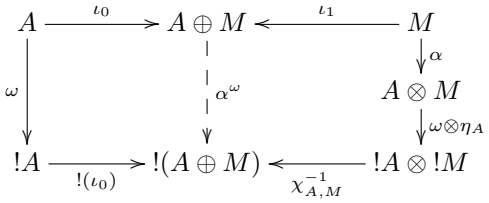
Alternatively using the additive structure, $\alpha^\omega := \pi_0 \omega !(\iota_0) + \pi_1 \alpha(\omega \otimes 1_M)(!(\iota_0) \otimes \iota_1)\mathsf{d}_{A \oplus M}$. Then $(A \oplus M, \alpha^\omega)$ is an $!$-coalgebra. Furthermore, the following equalities hold:

$$\Delta^{\alpha^\omega} := \pi_0 \Delta^\omega (\iota_0 \otimes \iota_0) + \pi_1 \alpha (\iota_0 \otimes \iota_1) + \pi_1 \sigma_{A,M} (\iota_1 \otimes \iota_0) \qquad \mathsf{e}^{\alpha^\omega} = \pi_0 \mathsf{e}^\omega$$

where $A \oplus M \xrightarrow{\Delta^{\alpha^\omega}} (A \oplus M) \otimes (A \oplus M)$ and $A \oplus M \xrightarrow{\mathsf{e}^{\alpha^\omega}} k$ are defined as in (6).

For a storage modality, α^ω can alternatively be defined using the Seely isomorphisms and codereliction:

▶ **Lemma 28.** *In a differential storage category with storage modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ *and codereliction* η*, if* (A, ω) *is a* $!$*-coalgebra and* (M, α) *a* (A, ω)*-comodule, then* α^ω *from the previous theorem can alternatively be described as the unique map which makes the following diagram commute:*



Alternatively using the additive structure, the following equality holds:

$$\alpha^\omega = \pi_0\omega!(\iota_0) + \pi_1\alpha(\omega \otimes \eta_M)\chi^{-1}_{A,M}$$

Proof. Using that $\mathsf{d}_A = (1_{!A} \otimes \eta_A)\nabla_A$ and $\chi^{-1}_{A,M} = (!(\iota_0) \otimes !(\iota_1))\nabla_{A\oplus M}$, we easily see that:

$$\begin{aligned}
\alpha^\omega &= \pi_0\omega!(\iota_0) + \pi_1\alpha(\omega \otimes 1_M)(!(\iota_0) \otimes \iota_1)\mathsf{d}_{A\oplus M} \\
&= \pi_0\omega!(\iota_0) + \pi_1\alpha(\omega \otimes 1_M)(!(\iota_0) \otimes \iota_1)(1_{!(A\oplus M)} \otimes \eta_{A\oplus M})\nabla_{A\oplus M} \\
&= \pi_0\omega!(\iota_0) + \pi_1\alpha(\omega \otimes \eta_M)(!(\iota_0) \otimes !(\iota_1))\nabla_{A\oplus M} \\
&= \pi_0\omega!(\iota_0) + \pi_1\alpha(\omega \otimes \eta_M)\chi^{-1}_{A,M}
\end{aligned}$$

So we conclude that the desired equality holds. ◀

We will now explain how $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ as constructed in Proposition 12 is of the form α^ω for a specific $!$-coalgebra and comodule. First observe that every object is a comodule of the monoidal unit, that is, for every object A, $(A, 1_A)$ is a $(k, 1_k, 1_k)$-comodule. Also note that it is easy to see that for the $!$-coalgebra (k, m_k), its associated comonoid is precisely $(k, 1_k, 1_k)$, that is, $\Delta^{\mathsf{m}_k} = 1_k$ and $\mathsf{e}^{\mathsf{m}_k} = 1_k$. Therefore, for every object A, $(A, 1_A)$ is a (k, m_k)-comodule.

▶ **Lemma 29.** *In a differential storage category with storage modality* $(!, \delta, \varepsilon, \Delta, \mathsf{e})$ *and codereliction* η*, for* $k \oplus A \xrightarrow{\mathsf{H}_A} !(k \oplus A)$ *as constructed in Proposition 12, the following equality holds:* $\mathsf{H}_A = 1_A^{\mathsf{m}_k}$*, where* $1_A^{\mathsf{m}_k}$ *is defined as in Theorem 27.*

Proof. Recall that $\mathsf{H}_A := \pi_0\mathsf{m}_k!(\iota_0) + \pi_1(\mathsf{m}_k \otimes \eta_A)\chi^{-1}_{k,A}$. By Lemma 28, since $\alpha = 1_A$ and $\omega = \mathsf{m}_k$, we clearly see that $\mathsf{H}_A = 1_A^{\mathsf{m}_k}$. ◀

▶ **Corollary 30.** H *satisfies* **[IA.1]***, that is, for every object* A*,* $(k \oplus A, \mathsf{H}_A)$ *is a* $!$*-coalgebra.*

Proof. This follows immediately from Theorem 27 and Lemma 29. ◀

It remains to prove **[IA.3]**, which we compute directly.

▶ **Lemma 31.** H *satisfies* **[IA.3]***, that is,* $(!A, \delta_A) \otimes^! (k \oplus B, \mathsf{H}_B) \xrightarrow{\Theta_{A,B}} (k \oplus (!A \otimes B), \mathsf{H}_{!A\otimes B})$ *is a* $!$*-coalgebra morphism.*

Proof. Recall that by construction $\Theta_{A,B} = (\mathsf{e}_A \otimes \pi_0)\iota_0 + (1_{!A} \otimes \pi_1)\iota_1$. We must show that $(\delta_A \otimes \mathsf{H}_B)\mathsf{m}_{!A,k\oplus B}!(\Theta_{A,B}) = \Theta_{A,B}\mathsf{H}_{!A\otimes B}$. By brute force computation, we show that:

$$
\begin{aligned}
&(\delta_A \otimes \mathsf{H}_B)\mathsf{m}_{!A,k\oplus B}!(\Theta_{A,B}) = \\
&= (\delta_A \otimes \pi_0)(1_{!!A} \otimes \mathsf{m}_k)(1_{!!A} \otimes !(\iota_0)\mathsf{m}_{!A,k\oplus B}!(\Theta_{A,B}) \\
&+ (\delta_A \otimes \pi_1)(1_{!!A} \otimes \mathsf{m}_k \otimes \eta_B)(1_{!!A} \otimes \chi^{-1}_{k,B})\mathsf{m}_{!A,k\oplus B}!(\Theta_{A,B}) \\
&= (\delta_A \otimes \pi_0)(1_{!A} \otimes \mathsf{m}_k)\mathsf{m}_{!A,k}!(1_{!A} \otimes \iota_0)!(\Theta_{A,B}) \\
&+ (\delta_A \otimes \pi_1)(1_{!!A} \otimes \mathsf{m}_k \otimes \eta_B)(\Delta_{!A} \otimes 1_{!k} \otimes 1_{!B})(1_{!!A} \otimes \sigma_{!!A,!k} \otimes 1_{!B}) \\
&(\mathsf{m}_{!A,!k} \otimes \mathsf{m}_{!A,!B})\,(!(1_{!A} \otimes \iota_0) \otimes !(1_{!A} \otimes \iota_1))\,\nabla_{!A\otimes(k\oplus B)}!(\Theta_{A,B}) \\
&= (\delta_A \otimes \pi_0)!(1_{!A} \otimes \iota_0)!(\Theta_{A,B}) \\
&+ (\delta_A \otimes \pi_1)(\Delta_{!A} \otimes \eta_B)(1_{!!A} \otimes \mathsf{m}_{!A,!B})\,(!(1_{!A} \otimes \iota_0) \otimes !(1_{!A} \otimes \iota_1))\,\nabla_{!A\otimes(k\oplus B)}!(\Theta_{A,B}) \\
&= (\delta_A \otimes \pi_0)!(\mathsf{e}_A)!(\iota_0) \\
&+ (\delta_A \otimes \pi_1)(\Delta_{!A} \otimes 1_B)(1_{!!A} \otimes \varepsilon_{!A} \otimes 1_B)(1_{!!A} \otimes \eta_{!A\otimes B}) \\
&(!(1_{!A} \otimes \iota_0) \otimes !(1_{!A} \otimes \iota_1))\,(!(\Theta_{A,B}) \otimes !(\Theta_{A,B}))\nabla_{k\oplus(!A\otimes B)} \\
&= (\mathsf{e}_A \otimes \pi_0)\mathsf{m}_k!(\iota_0) \\
&+ (\delta_A \otimes \pi_1)(\Delta_{!A} \otimes 1_B)(1_{!!A} \otimes \varepsilon_{!A} \otimes 1_B)(1_{!!A} \otimes \eta_{!A\otimes B})(!(\mathsf{e}_A) \otimes 1_{!(!A\otimes(k\oplus B))}) \\
&(!(\iota_0) \otimes !(\iota_1))\nabla_{k\oplus(!A\otimes B)} \\
&= (\mathsf{e}_A \otimes \pi_0)\mathsf{m}_k!(\iota_0) \\
&+ (\Delta_A \otimes \pi_1)(\delta_A \otimes \delta_A \otimes 1_B)(!(\mathsf{e}_A) \otimes \varepsilon_{!A} \otimes 1_B)(1_{!!A} \otimes \eta_{!A\otimes B})\chi^{-1}_{k,!A\otimes !A\otimes B} \\
&= (\mathsf{e}_A \otimes \pi_0)\mathsf{m}_k!(\iota_0) \\
&+ (\Delta_A \otimes \pi_1)(\mathsf{e}_A \otimes 1_{!A} \otimes 1_B)(\mathsf{m}_k \otimes \eta_{!A\otimes B})\chi^{-1}_{k,!A\otimes !A\otimes B} \\
&= (\mathsf{e}_A \otimes \pi_0)\mathsf{m}_k!(\iota_0) + (1_{!A} \otimes \pi_1)(\mathsf{m}_k \otimes \eta_{!A\otimes B})\chi^{-1}_{k,!A\otimes !A\otimes B} \\
&= \Theta_{A,B}\pi_0\mathsf{m}_k!(\iota_0) + \Theta_{A,B}\pi_1(\mathsf{m}_k \otimes \eta_! A \otimes B)\chi^{-1}_{k,!A\otimes B} \\
&= \Theta_{A,B}\mathsf{H}_{!A\otimes B}
\end{aligned}
$$

So we conclude that $\Theta_{A,B}$ is a !-coalgebra morphism and that H satisfies **[IA.3]**. ◀

So we conclude that H is indeed an infinitesimal extension.

B Proof of Proposition 13

By [3, Corollary 5], to show that a natural transformation $A \xrightarrow{\eta_A} !A$ is a codereliction it in fact suffices to show that η satisfies **[dC.3]**, **[dC.4]**, and **[dC.m]**. So let $k\oplus A \xrightarrow{\mathsf{H}_A} !(k\oplus A)$ be an infinitesimal extension and recall that $\eta_A : A \to !A$ is defined as follows $\eta_A := \iota_1\mathsf{H}_A!(\pi_1)$.

▶ **Lemma 32.** *η satisfies* **[dC.3]**.

Proof. We must show that $\eta_A\varepsilon_A = 1_A$. So we compute that:

$$\eta_A\varepsilon_A = \iota_1\mathsf{H}_A!(\pi_1)\varepsilon_A = \iota_1\mathsf{H}_A\varepsilon_{k\oplus A}\pi_1 = \iota_1\pi_1 = 1_A$$

So we conclude that η satisfies **[dC.3]**. ◀

▶ **Lemma 33.** *η satisfies* **[dC.4]**.

Proof. We must show that $\eta_A\delta_A = (\mathsf{u}_A \otimes \eta_A)(\delta_A \otimes \eta_{!A})\nabla_{!A}$. First note that we have the following equality:

$$\begin{aligned}
\mathsf{H}_A!(\pi_1) &= \pi_0\iota_0\mathsf{H}_A!(\pi_1) + \pi_1\iota_1\mathsf{H}_A!(\pi_1)\\
&= \pi_0\mathsf{m}_k!(\iota_0)!(\pi_1) + \pi_1\iota_1\mathsf{H}_A!(\pi_1)\\
&= \pi_0\mathsf{m}_k!(0) + \pi_1\iota_1\mathsf{H}_A!(\pi_1)\\
&= \pi_0\mathsf{u}_A + \pi_1\iota_1\mathsf{H}_A!(\pi_1)\\
&= \pi_0\mathsf{u}_A + \pi_1\eta_A
\end{aligned}$$

So $\mathsf{H}_A!(\pi_1) = \pi_0\mathsf{u}_A + \pi_1\eta_A$. Therefore, we compute:

$$\begin{aligned}
\eta_A\delta_A &= \iota_1\mathsf{H}_A!(\pi_1)\delta_A\\
&= \iota_1\mathsf{H}_A\delta_{k\oplus A}!!(\pi_1)\\
&= \iota_1\mathsf{H}_A!(\mathsf{H}_A)!!(\pi_1)\\
&= \iota_1\mathsf{H}_A!\left(\mathsf{H}_A!(\pi_1)\right)\\
&= \iota_1\mathsf{H}_A!\left(\pi_0\mathsf{u}_A + \pi_1\eta_A\right)\\
&= \iota_1\mathsf{H}_A\Delta_{k\oplus A}(!(\pi_0)\otimes !(\pi_1))(!(\mathsf{u}_A\otimes !(\eta_A))\nabla_{!A}\\
&= \iota_1\Lambda_A(\mathsf{H}_A\otimes\mathsf{H}_A)(!(\pi_0)\otimes !(\pi_1))(!(\mathsf{u}_A\otimes !(\eta_A))\nabla_{!A}\\
&= \iota_1\Lambda_A(\pi_0\otimes\mathsf{H}_A)(\mathsf{m}_k\otimes !(\pi_1))(!(\mathsf{u}_A\otimes !(\eta_A))\nabla_{!A}\\
&= \iota_1\mathsf{H}_A(\mathsf{u}_A\otimes !(\pi_1))(\delta_A\otimes !(\eta_A))\nabla_{!A}\\
&= (\mathsf{u}_A\otimes\eta_A)(\delta_A\otimes !(\eta_A))\nabla_{!A}\\
&= (\mathsf{u}_A\otimes\eta_A)(\delta_A\otimes\eta_{!A})\nabla_{!A}
\end{aligned}$$

So we conclude that η satisfies **[dC.4]**. ◀

▶ **Lemma 34.** *η satisfies* **[dC.m]**.

Proof. We must show that $(1_{!A}\otimes\eta_B)\mathsf{m}_{A,B} = (\varepsilon_A\otimes 1_B)\eta_{A\otimes B}$.

$$\begin{aligned}
(1_{!A}\otimes\eta_B)\mathsf{m}_{A,B} &= (1_{!A}\otimes\iota_1)(1_{!A}\otimes\mathsf{H}_B)(1_{!A}\otimes !(\pi_1))\mathsf{m}_{A,B}\\
&= (1_{!A}\otimes\iota_1)(\delta_A\otimes\mathsf{H}_B)(!(\varepsilon_A)\otimes !(\pi_1))\mathsf{m}_{A,B}\\
&= (1_{!A}\otimes\iota_1)(\delta_A\otimes\mathsf{H}_B)\mathsf{m}_{!A,k\oplus B}!\left(\varepsilon_A\otimes\pi_1\right)\\
&= (1_{!A}\otimes\iota_1)(\delta_A\otimes\mathsf{H}_B)\mathsf{m}_{!A,k\oplus B}!\left(1_{!A}\otimes\pi_1\right)!\left(\varepsilon_A\otimes 1_B\right)\\
&= (1_{!A}\otimes\iota_1)(\delta_A\otimes\mathsf{H}_B)\mathsf{m}_{!A,k\oplus B}!(\Theta_{A,B})!(\pi_1)!\left(\varepsilon_A\otimes 1_B\right)\\
&= (1_{!A}\otimes\iota_1)\Theta_{A,B}\mathsf{H}_{!A\otimes B}!(\pi_1)!\left(\varepsilon_A\otimes 1_B\right)\\
&= \iota_1\mathsf{H}_{!A\otimes B}!(\pi_1)!\left(\varepsilon_A\otimes 1_B\right)\\
&= \eta_{!A\otimes B}!\left(\varepsilon_A\otimes 1_B\right)\\
&= (\varepsilon_A\otimes 1_B)\eta_{A\otimes B}
\end{aligned}$$

So we conclude that η satisfies **[dC.m]**. ◀

So we conclude that η is a codereliction.

C Proof of Theorem 14

We must show that that constructions of Proposition 12 and Proposition 13 are inverses of each other. So starting with a codereliction η, we compute:

$$\iota_1\mathsf{H}_A!(\pi_1) = (\mathsf{m}_k\otimes\eta_A)\chi_{k,A}^{-1}!(\pi_1) = (\mathsf{m}_k\otimes\eta_A)(\mathsf{e}_k\otimes 1_{!A}) = \eta_A$$

Next starting with an infinitesimal extension H, in Lemma 33 we showed that $\mathsf{H}_A!(\pi_1) = \pi_0\mathsf{u}_A + \pi_1\eta_A$. Therefore, we compute that:

$$\begin{aligned}
\pi_0\mathsf{m}_k!(\iota_0) + \pi_1(\mathsf{m}_k \otimes \eta_A)\chi_{k,A}^{-1} &= \pi_0 \left(\mathsf{m}_k \otimes \mathsf{u}_A\right) \chi_{k,A}^{-1} + \pi_1 \left(\mathsf{m}_k \otimes \eta_A\right) \chi_{k,A}^{-1} \\
&= \pi_0\mathsf{u}_A \left(\mathsf{m}_k \otimes 1_{!A}\right) \chi_{k,A}^{-1} + \pi_1\eta_A \left(\mathsf{m}_k \otimes 1_{!A}\right) \chi_{k,A}^{-1} \\
&= \left(\pi_0\mathsf{u}_A + \pi_1\eta_A\right) \left(\mathsf{m}_k \otimes 1_{!A}\right) \chi_{k,A}^{-1} \\
&= \mathsf{H}_A!(\pi_1) \left(\mathsf{m}_k \otimes 1_{!A}\right) \chi_{k,A}^{-1} \\
&= \mathsf{H}_A \left(\mathsf{m}_k \otimes !(\pi_1)\right) \chi_{k,A}^{-1} \\
&= \Lambda_A(\pi_0 \otimes \mathsf{H}_A) \left(\mathsf{m}_k \otimes !(\pi_1)\right) \chi_{k,A}^{-1} \\
&= \Lambda_A(\mathsf{H}_A \otimes \mathsf{H}_A) \left(!(\pi_0) \otimes !(\pi_1)\right) \chi_{k,A}^{-1} \\
&= \mathsf{H}_A\Delta_{k\oplus A} \left(!(\pi_0) \otimes !(\pi_1)\right) \chi_{k,A}^{-1} \\
&= \mathsf{H}_A\chi_{k,A}\chi_{k,A}^{-1} \\
&= \mathsf{H}_A
\end{aligned}$$

So we conclude that coderelictions are in bijective correspondence with infinitesimal augmentations.

The Open Algebraic Path Problem

Jade Master
Department of Mathematics, University of California, Riverside, CA, USA

Abstract

The algebraic path problem provides a general setting for shortest path algorithms in optimization and computer science. We explain the universal property of solutions to the algebraic path problem by constructing a left adjoint functor whose values are given by these solutions. This paper extends the algebraic path problem to networks equipped with input and output boundaries. We show that the algebraic path problem is functorial as a mapping from a double category whose horizontal composition is gluing of open networks. We introduce functional open matrices, for which the functoriality of the algebraic path problem has a more practical expression.

2012 ACM Subject Classification Theory of computation → Categorical semantics; Theory of computation → Operational semantics

Keywords and phrases The Algebraic Path Problem, Open Systems, Shortest Paths, Categorical Semantics, Compositionality

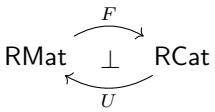
Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.20

Related Version *Available at*: `https://arxiv.org/abs/2005.06682`

Acknowledgements I would like to thank Mike Shulman, John Baez, Christian Williams, Joe Moeller, Rany Tith, Sarah Rovner-Frydman, Zans Mihejez, Oscar Hernandez, Alex Pokorny and Todd Trimble for their helpful comments and contributions. I would also like to thank everyone in my life who supported me during this time. Your work contributed to this paper as well. This paper was written on Tongva land.

1 Introduction

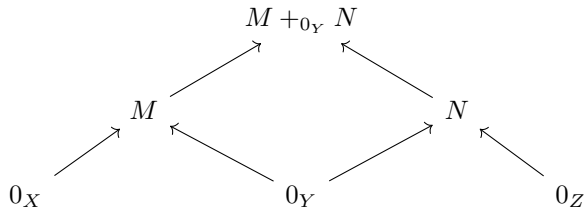
The algebraic path problem is a generalization of the shortest path problem to probability, computing, matrix multiplication, and optimization [18, 9]. Let $([0,\infty], \min, +)$ be the rig of positive real numbers with min as the "additive" monoid and $+$ as the "multiplicative" monoid. A matrix M valued in $[0,\infty]$ is regarded as a distance network and the shortest paths of M between all pairs of vertices may be computed using the geometric series formula: $F(M) = \sum_{n\geq 0} M^n$. The algebraic path problem frames many existing problems as generalizations of the shortest path problem by allowing $[0,\infty]$ to be replaced by a sufficiently nice rig R. Many popular shortest path algorithms can be extended to this more general setting [10] and the algebraic path problem can also be implemented generically using functional programming [5]. In Section 2, we show that finding solutions to the algebraic path problem can be understood as the left adjoint of an adjunction



between matrices valued in R and categories enriched in R.

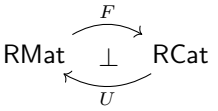
The algebraic path problem deals only with closed systems, i.e. systems which are isolated from their surroundings. On the other hand, open systems are equipped with input and output boundaries, from which they can be composed to form larger and more complicated networks. A research program intiated by Baez, Courser, and Fong aims to provide a theoretical foundation for open systems using cospan formalisms [7, 1]. For a category of

networks C, Baez and Courser defined a symmetric monoidal double category which provides a syntax for composition of open systems in C [1]. In Section 3, we set C equal to RMat, the category of matrices weighted in a quantale R, to obtain a symmetric monoidal double category $\mathsf{Open}(\mathsf{RMat})$. The essence of this double category is gluing. Open R-matrices are represented as cospans with feet given by 0-matrices. Given two such open R-matrices, take their pushout



to obtain an open R-matrix whose apex is synthesized from joining M and N along their shared boundary. This, along with the other data and structure of $\mathsf{Open}(\mathsf{RMat})$, provide a syntax for manipulating open R-matrices. The axioms of a symmetric monoidal double category guarantee that this syntax is well-behaved. For example, the word problem for double categories is solvable in quadratic time and double categories are equipped with a string diagram calculus [4, 14].

RCat, the category of R-enriched categories provides a choice of semantics for R-matrices, and can be expressed as R-matrices satisfying some regularity properties. In Section 2, we show how the solution to the algebraic path problem forms the left adjoint F of an adjunction below left



$$\bigstar \colon \mathsf{Open}(\mathsf{RMat}) \to \mathsf{Open}(\mathsf{RCat})$$

which provides a mapping from the syntax of R-matrices to the semantics of R-categories. R-categories equipped with input and output boundaries form the horizontal morphisms of a symmetric monoidal double category $\mathsf{Open}(\mathsf{RCat})$. In Section 4, we show how the algebraic path problem functor lifts to a symmetric monoidal double functor providing a coherent semantics for the syntax of *open* R-matrices as shown above right. This symmetric monoidal double functor provides a framework for studying how solutions to the algebraic path problem can be built inductively from gluings of smaller open R-matrices. The axioms of a symmetric monoidal double functor guarantee that this inductive process is well-behaved.

This result is more theoretical than practical. However, there is a subclass of open R-matrices, functional open R-matrices, for which the theory provides useful insight. Functional open R-matrices are roughly open R-matrices where the inputs are all sources and the outputs are all sinks. In Section 5 we show that there is a strict double functor

$$\blacksquare \circ \bigstar_{\mathrm{fxn}} \colon \mathsf{Open}(\mathsf{RMat})_{\mathrm{fxn}} \to \mathsf{Mat}_R$$

where Mat_R is a double category of R-matrices whose horizontal composition is matrix multiplication. This strict double functor gives a series of coherent compositional relationships for the algebraic path problem on functional open R-matrices based on matrix multiplication.

2 The Algebraic Path Problem

The algebraic path problem arises from the observation that various optimization problems can be framed in the same way by varying a sufficiently nice sort of rig. The level of generality for this work will be a commutative quantale, which is sufficient to guarantee existence and uniqueness of solutions to these optimization problems.

▶ **Definition 1.** *A* **quantale** *is a monoidal closed poset with all joins. Explicitly, a quantale is a poset R with an associative, unital, and monotone multiplication $\cdot\colon R \times R \to R$ such that for every index set I*

- *all joins, $\sum_{i\in I} x_i$, exist*
- *$\cdot$ preserves all joins, i.e.*

$$a \cdot \sum_{i\in I} x_i = \sum_{i\in I} a \cdot x_i.$$

A quantale is commutative if its multiplication operation, $\cdot$, is commutative.

A motivating example of such a quantale is the poset $[0,\infty]$ with $+$ as its monoidal product and with join given by infimum. Note that this poset is equipped with the reverse of the usual ordering on $[0,\infty]$. Fong and Spivak show how the shortest path problem on this quantale computes the shortest paths between all pairs of vertices in a given $[0,\infty]$-weighted graph [8, §2.5.3]. Other motivating examples include the rig $([0,1], \sup, \cdot)$ (whose algebraic path problem corresponds to most likely path in a Markov chain) and the powerset of the language generated by an alphabet (whose algebraic path problem corresponds to the language decided by a nondeterministic finite automata (NFA))[9].

▶ **Definition 2.** *For a commutative quantale R and sets X and Y, an* ***R*-matrix** *$M\colon X \to Y$ is a function $M\colon X \times Y \to R$. For R-matrices $M\colon X \to Y$ and $N\colon Y \to Z$, their matrix product MN is defined by the rule*

$$MN(i,k) = \sum_{j\in Y} M(i,j)N(j,k)$$

where juxtaposition denotes the multiplication of R.

If R is a commutative quantale, R-matrices form a quantale as well.

▶ **Definition 3.** *Let $\mathsf{RMat}(X)$ be the set of X-by-X matrices $M\colon X \times X \to R$. $\mathsf{RMat}(X)$ is equipped with the partial order $\leq$ where $M \leq N$ if and only if $M(i,j) \leq N(i,j)$ for all $i,j \in X$.*

▶ **Proposition 4.** *$\mathsf{RMat}(X)$ is a quantale with*

- *join given by pointwise sum of matrices,*
- *and multiplication given by matrix product.*

The proof of this proposition is left to the reader. All the required properties of $\mathsf{RMat}(X)$ follow from the analogous properties in R.

A square matrix $M\colon X \times X \to R$ represents a complete R-weighted graph whose vertex set is given by X.

▶ **Definition 5.** *Let $M\colon X \times X \to R$ be a square matrix. A* **vertex** *of M is an element $i \in X$. An* **edge** *of M is a tuple of vertices $(a,b) \in X \times X$. A* **path** *in M from a_0 to a_n is a list of adjacent edges $p = ((a_0,a_1),(a_1,a_2),\ldots,(a_{n-1},a_n))$. The* **weight** *of p is defined as the product $l(p) = \Pi_{i=0}^{n-1} M(a_i,a_{i+1})$ in R. For vertices $i,j \in X$, let $P_{ij}^M = \{$ paths in M from i to j $\}$*

Let i and j be vertices of a square matrix $M\colon X \times X \to R$. The algebraic path problem asks to compute the quantity $\sum_{p \in P_{ij}^M} l(p)$ in the quantale R. If R is the quantale $([0,\infty], \inf, +)$ then the weight of an edge M_{ij} represents the distance between vertex i and vertex j and the weight of a path $l(p)$ represents the total distance traversed by p. Summing the weights of all paths between a pair of vertices corresponds to finding the path with the minimum weight.

A more tractable framing of the algebraic path problem can be found by considering matrix powers. The entries of M^2 are given by

$$M^2(i,j) = \sum_{l \in X} M(i,l)M(l,j) = \inf_{l \in X}\{M(i,l) + M(l,j)\}.$$

Because $M(i,l)$ and $M(l,j)$ represent the distance from i to l and from l to j, this infimum computes the cheapest way to travel from i to j while stopping at some l in between. More generally, the entries of M^n for $n \geq 0$ represent the shortest paths between nodes of your graph that occur in exactly n steps. To compute the shortest paths which can occur in any number of steps, we must take the infimum of the matrices M^n over all $n \geq 0$. This pattern replicates for other choices of quantale. Therefore, the **algebraic path problem** seeks to compute

$$F(M) = \sum_{n \geq 0} M^n \tag{1}$$

where M is an R-matrix. The following table summarizes some instances of the algebraic path problem for different choices of R. Fink provides an explanation of the algebraic path problems for $([0,\infty], \leq)$ and $\{T, F\}$ and Foote provides an explanation for the quantales $([0,1], \leq)$ and $(\mathcal{P}(\Sigma), \subseteq)$ [6, 9].

poset	join	multiplication	solution of path problem
$([0,\infty], \geq)$	inf	$+$	shortest paths in a weighted graph
$([0,\infty], \leq)$	sup	inf	maximum capacity in the tunnel problem
$([0,1], \leq)$	sup	$\times$	most likely paths in a Markov process
$\{T, F\}$	OR	AND	transitive closure of a directed graph
$(\mathcal{P}(\Sigma^*), \subseteq)$	$\bigcup$	concatenation	decidable language of a NFA

Note that in this table, $\mathcal{P}(\Sigma^*)$ denotes the power set of the language generated by an alphabet Σ.

Equation (1) is known to category theorists by a different name: the free monoid on M. Framing it in this way gives a categorical proof of existence and uniqueness of $F(M)$. A classic result from [12, §V11] gives a construction of free monoids. MacLane's construction is defined as an adjunction into a category of internal monoids.

▶ **Definition 6.** *Let $(C, \otimes, I)$ be a monoidal category. A* **monoid internal to C** *is an object A of C equipped with morphisms*

$$m\colon A \otimes A \to A \text{ and } i\colon I \to A$$

satisfying the axioms of associativity and unitality expressed as commutative diagrams. A **monoid homomorphism** *from a monoid A to a monoid B is a morphism $f : A \to B$ in C which commutes with the maps m and i of each monoid. Let $\mathsf{Mon}(C)$ be the category where objects are monoids internal to C and morphisms are their homomorphisms.*

▶ **Proposition 7** (MacLane). *Let $(C, \otimes, I)$ be a monoidal category with countable coproducts such that tensoring on both sides preserves these coproducts. Then there is an adjunction below left*

$$C \underset{U}{\overset{F}{\rightleftarrows}} \mathsf{Mon}(C) \qquad F(X) = \sum_{n \geq 0} X^n$$

whose left adjoint is given by the countable coproduct of cartesian powers as shown above right.

The poset $\mathsf{RMat}(X)$ when viewed as a category satisfies the hypotheses of Proposition 7 and therefore admits a free monoid construction.

▶ **Proposition 8.** *There is an adjoint pair*

$$\mathsf{RMat}(X) \underset{U_X}{\overset{F_X}{\rightleftarrows}} \mathsf{Mon}(\mathsf{RMat}(X))$$

where F_X is the monotone map which produces the solution to the algebraic path problem on a matrix and U_X is the natural forgetful map.

Proof. Because $\mathsf{RMat}(X)$ is a quantale, it can be regarded as a monoidal category with all coproducts such that tensoring distributes over these coproducts. The result follows from applying Proposition 7 and noticing that MacLane's construction of free monoids matches Equation 1 in the case when $C = \mathsf{RMat}(X)$. ◀

Monoids internal to $\mathsf{RMat}(X)$ are R-enriched categories.

▶ **Definition 9.** *An R-category C with object set X consists of an element $C(x, y)$ in R for every $x, y \in X$ such that*

- *$1 \leq C(x, x)$ (the identity law), and*
- *$C(x, y)C(y, z) \leq C(x, z)$ (the composition law).*

Let $\mathsf{RCat}(X)$ be the poset whose elements are R-enriched categories with object set X. For R-categories C and D,

$$C \leq D \leftrightarrow C(i, j) \leq D(i, j) \quad \forall i, j \in X$$

▶ **Proposition 10.** $\mathsf{Mon}(\mathsf{RMat}(X))$ *is isomorphic to* $\mathsf{RCat}(X)$*, the poset of categories enriched in R with object set X.*

Proof. The isomorphism in question assigns a matrix $M \colon X \times X \to R$ to the R-category with $\hom(x, y) = M(x, y)$. The identity law follows from the inequality $1 \leq M$ and the inequality $M^2 \leq M$ implies that for all $y \in X$, $\sum_{y \in X} M(x, y)M(y, z) \leq M(x, z)$. The composition law follows from the fact that any element of R is less than a join which contains it. ◀

Proposition 8 says that each matrix valued in R has a unique, universally characterized solution to the algebraic path problem: namely the free R-category on that matrix. This adjunction can be extended to matrices over an arbitrary set.

▶ **Definition 11.** *Let $f : X \to Y$ be a function and let $M : X \times X \to R$ be an R-matrix. Then the* **pushforward** *of M along f is the matrix $f_*(M) \colon Y \times Y \to R$ defined by*

$$f_*(M)(y, y') = \sum_{(x,x') \in (f \times f)^{-1}(y,y')} M(x, x').$$

▶ **Definition 12.** *Let* RMat *be the category where objects are square matrices $M \colon X \times X \to R$ on some set X and where a morphism of R-matrices from $M \colon X \times X \to R$ to $N \colon Y \times Y \to R$ is a function $f \colon X \to Y$ satisfying $f_*(M) \leq N$. Let* RCat *be the full subcategory of* RMat *consisting of matrices satisfying the axioms of an R-category (see Definition 9).*

The above adjunction may be extended to square matrices over an arbitrary set. We leave the proof of the following proposition to Appendix A.

▶ **Proposition 13.** *The free monoid construction of Proposition 8 extends to an adjunction*

$$\mathsf{RMat} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{RCat}.$$

The following proposition will be useful in the next section.

▶ **Proposition 14.** *The above adjunction $F \dashv U$ is idempotent.*

Proof. Every adjunction between posets is idempotent. Therefore the smaller adjunctions $F_X \dashv U_X$ are idempotent. Because F and U are stitched together using these adjunctions, it is idempotent as well. ◀

3 Open R-Matrices

R-matrices are made open by designating some of their vertices to be either inputs or outputs. In this section we show how these open R-matrices are composed by joining the output vertices of one to the input vertices of another and joining the data on the overlap. To define open R-matrices, we need a notion of a discrete R-matrix on a set X i.e. a matrix whose entries are all zero. The map sending a set to its discrete R-matrix is a functor and a left adjoint.

▶ **Proposition 15.** *Let $R \colon \mathsf{RMat} \to \mathsf{Set}$ be the functor which sends an R-matrix to its underlying set of vertices and sends a morphism to its underlying function. Then U has a left adjoint*

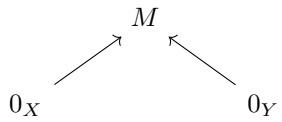
$$0 \colon \mathsf{Set} \to \mathsf{RMat} \quad 0_X \colon X \times X \to Y$$

which sends a set X to the R-matrix defined by $0_X(i, j) = 0$ for all i and j in X. F sends a function $f \colon X \to Y$ to the morphism of R-matrices which has f as its underlying function between vertices.

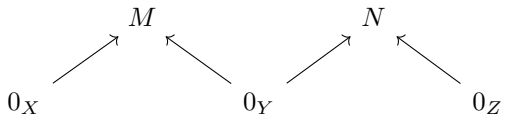
Proof. The natural isomorphism $\phi \colon \mathsf{RMat}(0_X, G) \cong \mathsf{Set}(X, R(G))$ is formed by noting that a morphism $0_X \to R(G)$ is uniquely determined by its underlying function on vertices and every such function obeys the inequality in Definition 12. ◀

A weighted graph can be opened up to its environment by equipping it with inputs and outputs.

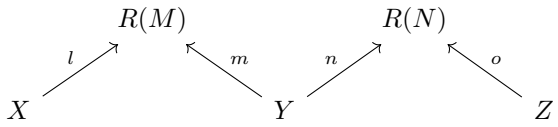
▶ **Definition 16.** *Let $M : A \times A \to R$ be an R-matrix. An* **open R-matrix** $M \colon X \to Y$ *is a cospan in* RMat *of the form*



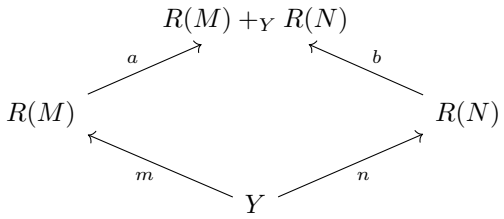
The idea is that the maps of this cospan point to input and output nodes of the matrix M. Let $M \colon X \to Y$ and $N \colon Y \to Z$



be open R-matrices. The underlying sets of M and N form a diagram



which generate a pushout



The functions a and b of this pushout allow the matrices M and N to be compared on equal footing: the pushforwards $a_*(M)$ and $b_*(N)$ both have $R(M) +_Y R(N)$ as their underlying set. The matrices $a_*(M)$ and $b_*(N)$ are combined using pointwise sum.

▶ **Definition 17.** *For open R-matrices $M : X \to Y$ and $N \colon Y \to Z$ as defined above, their* **composite** *is defined by*

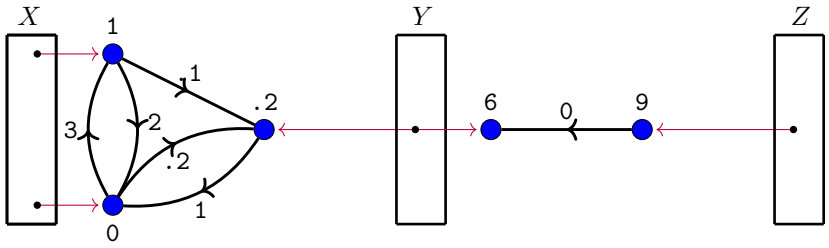
$$N \circ M \colon X \to Z = \quad \begin{array}{ccccc} & & a_*(M) + b_*(N) & & \\ & {\scriptstyle \phi^{-1}(a \circ l)} \nearrow & & \nwarrow {\scriptstyle \phi^{-1}(b \circ r)} & \\ 0_X & & & & 0_Z \end{array}$$

where ϕ^{-1} gives the unique morphism out of a discrete R-matrix defined by a function on its underlying set.

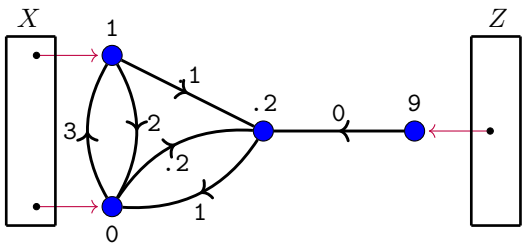
An R-matrix $M \colon X \times X \to R$ can represent a graph with vertex set X weighted in R. Similarly, an open R-matrix, represents an R-weighted graph equipped with inputs and outputs. For example, the $[0, \infty]$-matrices on the sets $\{a, b, c\}$ and $\{d, e\}$

$$\begin{bmatrix} 1 & 2 & .1 \\ 3 & 0 & .2 \\ \infty & 1 & .2 \end{bmatrix} \qquad \begin{bmatrix} 6 & \infty \\ 0 & 9 \end{bmatrix}$$

respectively may be upgraded to open $[0,\infty]$-matrices as follows. The first matrix has input set $\{1,2\}$ and output set $\{3\}$. The mappings of the cospan are given by $1 \mapsto a, 2 \mapsto b$ and $3 \mapsto c$. Similarly, the second matrix has left input set given by $\{3\}$ and right input set given by $\{4\}$. The mappings in the cospan for this open $[0,\infty]$-matrix are given by the assignments $3 \mapsto d$ and $4 \mapsto e$. These two open $[0,\infty]$-matrices are drawn as follows:



In this picture, edges are ommitted when their value is ∞ and a label on a vertex indicates the weight of the edge from that vertex to itself. These two open $[0,\infty]$-matrices are composed by identifying vertices mapped to by a common element of Y.



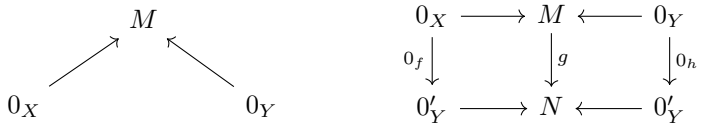
where edges are omitted if their weight is infinite in both directions. The matrix on the apex of this composite is computed by pushing each component matrix forward to the pushout of their underlying sets and adding them together i.e.

$$\begin{bmatrix} 1 & 2 & .1 & \infty \\ 3 & 0 & .2 & \infty \\ \infty & 1 & .2 & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} + \begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & 6 & \infty \\ \infty & \infty & 0 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & .1 & \infty \\ 3 & 0 & .2 & \infty \\ \infty & 1 & .2 & \infty \\ \infty & \infty & 0 & 9 \end{bmatrix}$$

where $+$ denotes the pointwise join of the quantale $[0,\infty]$. The entries of this matrix represent the shortest distances between pairs of vertices in the composite open $[0,\infty]$-matrix. This composition forms the horizontal composition of a symmetric monoidal double category. Note that the double categories considered here are called pseudo–double categories.

▶ **Theorem 18.** *For a quantale R, there is a symmetric monoidal double category* $\mathsf{Open}(\mathsf{RMat})$ *where*

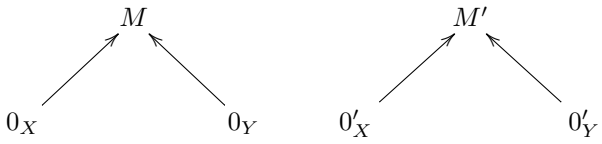
- *objects are sets X,Y,$Z\ldots$*
- *vertical morphisms are functions $f : X \to Y$,*
- *a horizontal morphism $M\colon X \to Y$ is an open R-matrix below left*



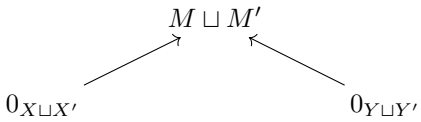
- *vertical 2-morphisms are commutative rectangles shown above right,*
- *vertical composition is ordinary composition of functions,*
- *and horizontal composition is given by the composite operation defined above.*

The symmetric monoidal structure is given by

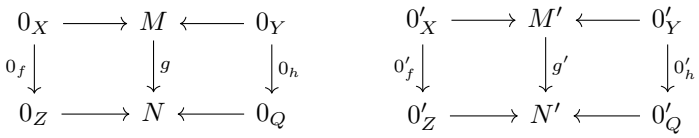
- *coproducts in* Set *on objects and vertical morphisms,*
- *pointwise coproducts on horizontal morphisms i.e. for open R-matrices,*



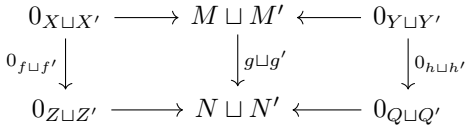
their coproduct is



and pointise coproduct for two vertical 2-morphisms i.e. for vertical 2-morphisms,



their coproduct is



Proof. Theorem 3.2.3 of [3] constructs this symmetric monoidal double category as long as

- RMat has coproducts and pushouts, and
- $0\colon \mathsf{Set} \to \mathsf{RMat}$ preserves pushouts and coproducts.

Because 0 is a left adjoint (Proposition 15) it preserves pushouts and coproducts when they exist so it suffices to prove the following lemma which we do in Appendix A. ◀

▶ **Lemma 19.** RMat *has coproducts and pushouts.*

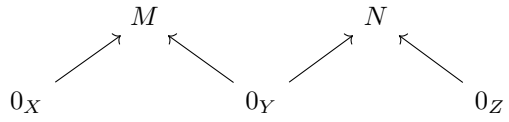
4 Compositionality of the Algebraic Path Problem

In this section we show how the algebraic path problem functor $F\colon \mathsf{RMat} \to \mathsf{RCat}$ extends to a symmetric monoidal double functor

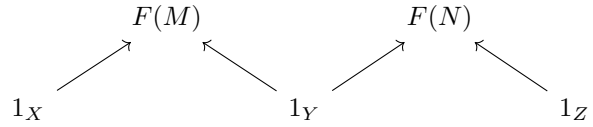
$$\mathsf{Open}(F)\colon \mathsf{Open}(\mathsf{RMat}) \to \mathsf{Open}(\mathsf{RCat}).$$

This double functor describes how the syntax of gluing open R-matrices extends to a series of coherent compositionality laws for the algebraic path problem.

For composable open R-matrices



we may apply the algebraic path problem functor F to the entire diagram to get cospans of R-categories



where 1_X is the identity matrix on X with respect to matrix multiplication. The pushout in RMat, $F(M) +_{1_Y} F(N)$, is not equal to the solution $F(M +_{0_Y} N)$. The former optimizes only over paths which start in M and end in N. On the other hand, $F(M +_{0_Y} N)$ optimizes over paths which may zig-zag back and forth between M and N, as many times as they like, before arriving at their destination. Therefore, to construct $F(M +_{0_Y} N)$ from its components we turn to the pushout in RCat.

▶ **Proposition 20.** RCat *has pushouts and coproducts.*

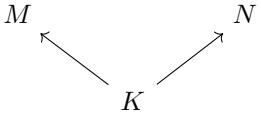
Proof. More generally, RCat has all colimits by Corollary 2.14 of [19]. These colimits are constructed via the transfinite construction of free algebras [11]. The idea behind the transfinite construction is that colimits in a category of monoids can be constructed by first taking the colimit of their underlying objects, taking the free monoid on that colimit, and then quotienting out by the equations in your original monoids. Next we provide an explicit description in the case of R-categories. ◀

▶ **Proposition 21.** *For a diagram* $D\colon C \to \mathsf{RCat}$*, its colimit is given by the formula*

$$\mathrm{colim}_{c\in C} D(c) \cong F(\mathrm{colim}_{c\in C} U(D(c)))$$

Proof. It suffices to show that $F(\mathrm{colim}_{c\in C} U(D(c)))$ satisfies the universal property of $\mathrm{colim}_{c\in C} D(c)$. Let $\alpha\colon \Delta_d \Rightarrow D$ be a cocone from an object $d \in \mathsf{RCat}$ to our diagram D. Because α can be regarded as a cocone in RMat, the universal property of colimits induces a unique map $\mathrm{colim}_{c\in C} U(D(c)) \to U(d)$ of R-matrices. Applying F to this morphism gives a map $F(\mathrm{colim}_{c\in C} U(D(c))) \to FU(d) = d$ where the last equality follows from the adjunction $F \dashv U$ being idempotent as shown in Proposition 14. The above map is a unique morphism satsifying the universal property for $\mathrm{colim}_{c\in C} D(c)$. ◀

▶ **Corollary 22.** *For a diagram below left*



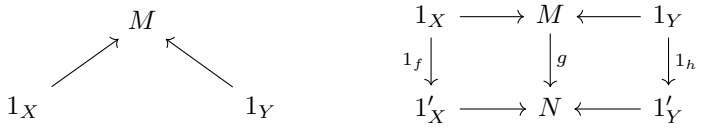
$$M +_K N \cong F(U(M) +_{U(K)} U(N))$$

in RCat*, the pushout is given by the isomorphism above right.*

This pushout forms the horizontal composition of a double category of open R-categories.

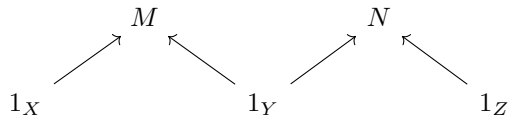
▶ **Theorem 23.** *There is a symmetric monoidal double category* $\mathsf{Open}(\mathsf{RCat})$ *where*

- *objects are sets,*
- *vertical morphisms are functions,*
- *horizontal morphisms are cospans shown below left*

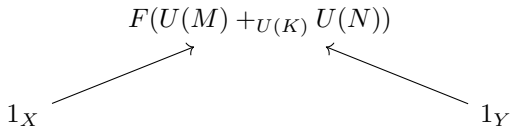


where the apex M *satisfies the axioms of an* R*-category, and*

- *vertical 2-morphisms are commuting rectangles shown above right*
- *The horizontal composition is given by pushout of open* R*-categories i.e. for open* R*-categories*



their pushout is the cospan

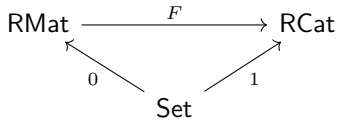


The symmetric monoidal structure of $\mathsf{Open}(\mathsf{RCat})$ *is given by*

- *coproduct of sets and functions,*
- *pointwise coproduct on horizontal morphisms,*
- *and pointiwise coproduct on vertical 2-morphisms.*

Proof. To construct the desired symmetric monoidal double category, we apply Corollary 2.4 of [1] to the composite left adjoint $\mathsf{Set} \xrightarrow{0} \mathsf{RMat} \xrightarrow{F} \mathsf{RCat}$. ◀

So far we have the commutative diagram of functors



where $1\colon \mathsf{Set} \to \mathsf{RCat}$ is the functor which sends a set X to the identity matrix 1_X. The definition of Open is functorial with respect to this sort of diagram i.e. it induces a symmetric monoidal double functor between the relevant double categories.

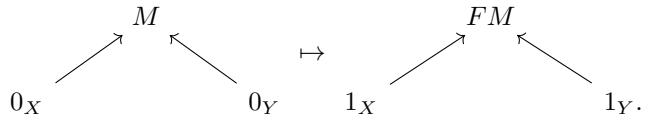
▶ **Theorem 24.** *There is a symmetric monoidal double functor*

$$\bigstar\colon \mathsf{Open}(\mathsf{RMat}) \to \mathsf{Open}(\mathsf{RCat})$$

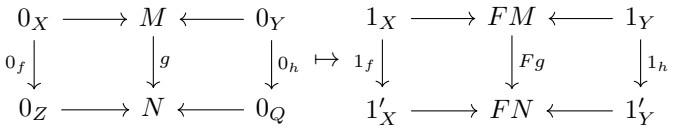
which is

- *the identity on objects and vertical morphisms,*

- *sends an open R-matrix* $M\colon X \to Y$ *below left*

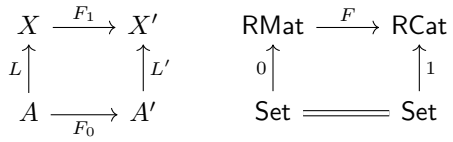


to the solution of its algebraic path problem $\bigstar(M)\colon X \to Y$ *above right, and*
- *a vertical 2-morphism of open R-matrices* $\alpha\colon M \Rightarrow N$ *below left*



is sent to the 2-morphism $\bigstar(\alpha)\colon M \Rightarrow N$ *above right given by pointwise application of* F*.*

Proof. Theorem 4.3 of [1] proves functoriality of the "Open" construction on squares below left



commuting up to natural isomorphism. The result follows from applying this result to the square shown above right. ◀

The definition of symmetric monoidal double functor packages up a lot of information very succinctly. In particular, it contains coherent comparison isomorphism relating the solution of the algebraic path problem on a composite matrix to the solution on its components. For open R-matrices $M : X \to Y$ and $N : Y \to Z$, there is a composition comparison

$$\phi_{MN}\colon \bigstar(M) \circ \bigstar(N) \xrightarrow{\sim} \bigstar(M \circ N) \tag{2}$$

and monoidal comparison

$$\psi_{MM'}\colon \bigstar(M + M') \xrightarrow{\sim} \bigstar(M) + \bigstar(M') \tag{3}$$

giving recipes to break solutions to the algebraic path problem into their components. In other words, the left-hand side of each comparison is computed to determine the right-hand side

Pouly and Kohlas present a similar relationship in the context of valuation algebras. [15, §6.7]. For matrices M and N representing weighted graphs on vertex sets s and t respectively, the solution to the algebraic path problem on the union of their vertex sets is given by

$$F(M) \otimes F(N) = F\left(F(M)^{\uparrow s\cup t} + F(N)^{\uparrow s\cup t}\right)$$

In this formula, $\uparrow s \cup t$ indicates that the matrix is trivially extended to the union of the vertex sets. This formula is less general than comparison (2): it corresponds to the special case when the legs of the open R-matrices are inclusions.

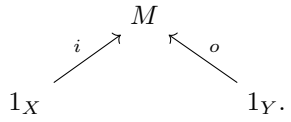
A typical algorithm for the algebraic path problem has spacial complexity $\Theta(n^3)$ where n is the number of vertices in your weighted graph [10]. The comparisons (2) and (3) suggest a strategy for computing the solution to the algebraic path problem which reduces this

complexity. First cut your weighted graph into smaller chunks, compute the solution to the algebraic path problem on those chunks, then combine their solutions using (2) and 3). Unfortunately, this strategy will in general take *more* time to compute the solution to the algebraic path problem on a composite because the right hand side of comparison (2) requires three applications of the functor F. However, the situation improves if the open R-matrices are functional.

5 Functional Open Matrices

In this section we define functional open R-matrices, a class of open R-matrices for which the composition comparison $\phi_{MN} \colon \bigstar(M) \circ \bigstar(N) \cong \bigstar(M \circ N)$ can be expressed in terms of matrix multiplication. The one caveat is that this expression requires that the open matrices be restricted to their inputs and outputs. For this we borrow a concept from engineering called "blackboxing" which forgets the internal workings of a system and concentrates only on the relationship it induces between its inputs and outputs.

▶ **Definition 25.** *Let $M \colon X \to Y$ be the open R-category*



Then the **blackboxing** *of M is the matrix below left*

$$\blacksquare(M) \colon X \times Y \to R \qquad \blacksquare(M)(x,y) = M(i(x), o(y))$$

given by the expression above right.

The $\blacksquare$ operation is extended to all of RCat but the composition is only preserved laxly. The codomain of this extension is the following:

▶ **Definition 26.** *Let Mat_R be the double category where*

- *an object is a set X,Y,Z,...*
- *a vertical morphism is a function $f \colon X \to Y$,*
- *a horizontal morphism $M \colon X \to Y$ is a matrix $M \colon X \times Y \to R$,*
- *a vertical 2-morphism from $M \colon X \to Y$ to $N \colon X' \to Y'$ is a square below left*

$$\begin{array}{ccc} X & \xrightarrow{M} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[N]{} & Y' \end{array} \qquad \sum_{x \in f^{-1}(x'),\, y \in g^{-1}(y')} M(x,y) \leq N(x',y')$$

  satisfing the inequality above right for all $x' \in X'$ and $y' \in Y'$.
- *Vertical composition is function composition,*
- *and horizontal composition is given by matrix multiplication.*

In this double category, the composite of matrices M and N is written as the juxtaposition MN. Blackboxing is extended to the double category of open R-categories.

▶ **Proposition 27.** *There is a lax double functor*

$$\blacksquare \colon \mathsf{Open}(\mathsf{RCat}) \to \mathsf{Mat}_R$$

which

- *is the identity on objects,*
- *sends an open R-category $M\colon X \to Y$ to its blackbox $\blacksquare(M)$,*
- *and sends a vertical 2-cell below left*

$$\begin{array}{ccccc} 1_X & \longrightarrow & M & \longleftarrow & 1_Y \\ {\scriptstyle 1_f}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle 1_h} \\ 1_{X'} & \longrightarrow & N & \longleftarrow & 1_{Y'} \end{array} \quad \mapsto \quad \begin{array}{ccc} X & \xrightarrow{\blacksquare(M)} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[\blacksquare(N)]{} & Y' \end{array}$$

to the vertical 2-cell above right.

The blackboxing functor is composed with the algebraic path problem functor to get a lax symmetric monoidal double functor

$$\mathsf{Open}(\mathsf{RMat}) \xrightarrow{\bigstar} \mathsf{Open}(\mathsf{RCat}) \xrightarrow{\blacksquare} \mathsf{Mat}_R$$

This lax symmetric monoidal double functor sends an open R-matrix to the solution of its algebraic path problem only on nodes which start with an input and end with an output. It is natural to ask when this mapping is strictly functorial, as this yields a simple compositional formula for the algebraic path problem:

$$\blacksquare(\bigstar(M \circ N)) = \blacksquare(\bigstar(M))\blacksquare(\bigstar(N)).$$

The double functor $\blacksquare \circ \bigstar$ is strictly functorial on functional open matrices.

▶ **Definition 28.** *Let $M\colon A \times A \to R$ be an R-matrix. An element $a \in X$ is a* **source** *if for every $b \in X$, $M(b,a) = 0$ and a* **sink** *if $M(a,b) = 0$. A* **functional open R-matrix** *is an open R-matrix*

$$\begin{array}{ccccc} & & M & & \\ & {\scriptstyle l}\nearrow & & \nwarrow{\scriptstyle r} & \\ 0_X & & & & 0_Y \end{array}$$

such that for every $x \in X$, $l(x)$ is a source and for every $y \in Y$, $r(y)$ is a sink.

Because the pushout of functional open R-matrices is also functional, we can form the following sub-double category.

▶ **Definition 29.** *Let $\mathsf{Open}(\mathsf{RMat})_{fxn}$ be the full sub-symmetric monoidal double category generated by the open R-matrices which are functional.*

▶ **Theorem 30.** *The composite $\blacksquare \circ \bigstar$ restricts to a strict double functor*

$$\blacksquare \circ \bigstar_{fxn}\colon \mathsf{Open}(\mathsf{RMat})_{fxn} \to \mathsf{Mat}_R$$

The proof of this theorem relies on a lemma which resembles the the binomial expansion of $(a+b)^n$ in a ring where $ba = 0$. If a and b represent blackboxings of functional open matrices, then the identity $ba = 0$ indicates that there are no paths which go backwards.

▶ **Lemma 31.** *For functional open R-matrices $M\colon X \to Y$ and $N\colon Y \to Z$ we have that*

$$\blacksquare(M +_{1_Y} N)^n = \sum_{i+j=n} \blacksquare(M^i)\blacksquare(N^j)$$

Proof. The entries of the left hand side are expanded as

$$\blacksquare((M +_{1_Y} N)^n)(a_0, a_n) = \sum_{a_1,\cdots,a_{n-1}} (M +_{1_Y} N)(a_0, a_1)(M +_{1_Y} N)(a_1, a_2) \cdots (M +_{1_Y} N)(a_{n-1}, a_n)$$

where the a_i are equivalence classes in $RM +_Y RN$. For a particular term of this sum, let $1 \leq k \leq n$ be the first natural number such that a_k contains an element of RN. Because M and N are functional, for $k \leq i \leq n$ the equivalence classes a_i must also contain an element of RN if our term is nonzero. Therefore for a fixed k the contribution to the above sum is given by

$$\sum M(a_0, a_1) \dots M(a_{k-1}, a_k) N(a_k, a_{k+1}) \dots N(a_{n-1}, a_n)$$

which simplifies to

$$\blacksquare(M^k)\blacksquare(N^{n-k})(a_0, a_n).$$

Because k can occur in any entry we have that

$$\blacksquare((M +_{1_Y} N)^n) = \sum_{k \leq n} \blacksquare(M^k)\blacksquare(N^{n-k}) = \sum_{i+j=n} \blacksquare(M^i)\blacksquare(N^j)$$

and this completes the proof. ◀

Proof of Theorem 30: It suffices to prove that for functional open R-matrices $0_X \longrightarrow M \longleftarrow 0_Y$ and $0_Y \longrightarrow N \longleftarrow 0_Z$ the equation

$$\blacksquare(\bigstar(M \circ N)) = \blacksquare(\bigstar(M))\blacksquare(\bigstar(N))$$

holds. Consider the left-hand side:

$$\begin{aligned} \blacksquare\bigstar M \circ N &= \blacksquare \sum_{n \geq 0} (M \circ N)^n \\ &= \sum_{n \geq 0} \blacksquare(M \circ N)^n \\ &= \sum_{n \geq 0} \sum_{i+j=n} \blacksquare(M^i)\blacksquare(N^j) \end{aligned}$$

on the other hand,

$$\begin{aligned} \blacksquare(\bigstar(M))\blacksquare(\bigstar(N)) &= \sum_{i \geq 0} \blacksquare(M^i) \sum_{j \geq 0} \blacksquare(N^j) \\ &= \sum_{i,j \geq 0} \blacksquare(M^i)\blacksquare(M^j) \end{aligned}$$

Both sums contain the term $\blacksquare(M^i)\blacksquare(N^j)$ for every value of i and j, but the left hand side may contain repeated terms. However, because addition is idempotent, repeated terms don't contribute to the sum and the two sides are the same. ◀

6 Conclusion

The functoriality of Theorem 30 might not be surprising. It says that if your open matrices are joined together directionally along bottlenecks, then the computation of the algebraic path problem can be reduced to a computation on components. This strategy has already proven sucessful. In [17], Sairam, Tamassia, and Vitter show how choosing *one way separators* as cuts in a graph, allow for an efficient divide and conquer parallel algorithm for computing shortest paths. In [16] Rathke, Sobocinksi, and Stephens show how the reachability problem on a 1-safe Petri net can be computed more efficiently by cutting it up into more manageable pieces. Theorem 24 provides a framework for compositional formulas of this type. In future work we plan on extending the construction of this theorem to many other sorts of discrete event dynamic systems.

Lemma 31 also holds independent computational interest. The equation given there gives a novel compositional formula for computing the solution to the algebraic path problem. The author has implemented this formula for the special case of Markov processes [13]. We hope that this is the start of a more extensive library, made faster and more reliable by the mathematics developed in this paper.

References

1 John C Baez and Kenny Courser. Structured cospans. *Theory and Applications of Categories*, 35(48):1771–1822, 2020.

2 Francis Borceux. *Handbook of Categorical Algebra: Volume 1, Basic Category Theory*. Cambridge University Press, 1994.

3 Kenny Courser. *Open Systems: a Double Categorical Perspective*. PhD thesis, University of California Riverside, 2020.

4 Antonin Delpeuch. The word problem for double categories. *Theory and Applications of Categories*, 35(1):1–18, 2020.

5 Stephen Dolan. Fun with semirings: a functional pearl on the abuse of linear algebra. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*, pages 101–110, 2013.

6 Eugene Fink. *A Survey of Sequential and Systolic Algorithms for the Algebraic Path Problem*. University of Waterloo, Department of Mathematics, 1992.

7 Brendan Fong. *The Algebra of Open and Interconnected Systems*. PhD thesis, University of Oxford, 2016.

8 Brendan Fong and David I Spivak. *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press, 2019.

9 Davis Foote. Kleene algebras and algebraic path problems. 2015. Available at edge.edx.org.

10 Peter Höfner and Bernhard Möller. Dijkstra, Floyd and Warshall meet Kleene. *Formal Aspects of Computing*, 24(4-6):459–476, 2012.

11 G Max Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bulletin of the Australian Mathematical Society*, 22(1):1–83, 1980.

12 Saunders Mac Lane. *Categories for the Working Mathematician*. Springer Science & Business Media, 2013.

13 Jade Master. Compositional markov. https://github.com/Jademaster/compositionalmarkov, 2020.

14 David Jaz Myers. String diagrams for double categories and equipments. 2016. Available at https://arxiv.org/abs/1612.02762.

15 Marc Pouly and Jürg Kohlas. *Generic Inference: a Unifying Theory for Automated Reasoning*. John Wiley & Sons, 2012.

16 Julian Rathke, Paweł Sobociński, and Owen Stephens. Compositional reachability in Petri nets. In *International Workshop on Reachability Problems*, pages 230–243. Springer, 2014.

17 Sairam Subramanian, Roberto Tamassia, and Jeffrey Scott Vitter. An efficient parallel algorithm for shortest paths in planar layered digraphs. *Algorithmica*, 14(4):322–339, 1995.

18 Robert Endre Tarjan. A unified approach to path problems. *Journal of the Association for Computing Machinery*, 28(3):577–593, 1981.

19 Harvey Wolff. V-cat and V-graph. *Journal of Pure and Applied Algebra*, 4(2):123–135, 1974.

A Ommitted Proofs

▶ **Proposition 32.** *The free monoid construction of Proposition 8 extends to an adjunction*

$$\mathsf{RMat} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{RCat}.$$

Proof. Let $A\colon \mathsf{Set} \to \mathsf{Cat}$ be the functor which sends a set X to the poset $\mathsf{RMat}(X)$ regarded as a category and sends a function $f\colon X \to Y$ to the pushforward functor

$$f_*\colon \mathsf{RMat}(X) \to \mathsf{RMat}(Y).$$

from Definition 11. Analogously, let $B\colon \mathsf{Set} \to \mathsf{Cat}$ be the functor which sends a set X to the poset $\mathsf{RCat}(X)$ and sends a function f to its pushforward functor. The functors F_X for each set X form the components of a natural transformation $\mathbf{F}\colon A \Rightarrow B$. Similarly, the functors U_X form the components of a natural transformation $\mathbf{U}\colon B \Rightarrow A$. Furthermore, these natural transformations form an adjoint pair in the 2-category $[\mathsf{Set}, \mathsf{Cat}]$ of functors $\mathsf{Set}^{\mathrm{op}} \to \mathsf{Cat}$, natural transformations between them, and modifications. $\mathbf{F}$ and $\mathbf{U}$ are adjoint because an adjoint pair in $[\mathsf{Set}, \mathsf{Cat}]$ is the same as a pair of natural transformations which are adjoint in each component. To summarize, we have a pair of adjoint natural transformations as follows:

$$\mathsf{Set} \underset{B}{\overset{A}{\underset{\Downarrow}{\overset{\Uparrow}{\mathbf{F} \dashv \mathbf{U}}}}} \mathsf{Cat}$$

A restriction of the Grothendieck construction [2] defines a 2-functor

$$\int : [\mathsf{Set}, \mathsf{Cat}] \to \mathsf{CAT}$$

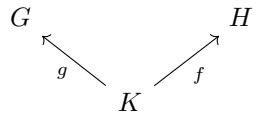
where CAT is the 2-category of large categories. Because every 2-functor preserves adjunctions, the above diagram maps to an adjunction

$$\int A \underset{\int \mathbf{U}}{\overset{\int \mathbf{F}}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \int B.$$

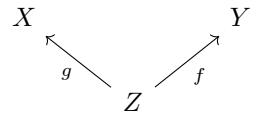
The result follows from the equivalences $\int A \cong \mathsf{RMat}$ and $\int B \cong \mathsf{RCat}$. The desired functors F and U are obtained by composing $\int \mathbf{F}$ and $\int \mathbf{U}$ with these equivalences. ◀

▶ **Lemma 33.** RMat *has coproducts and pushouts.*

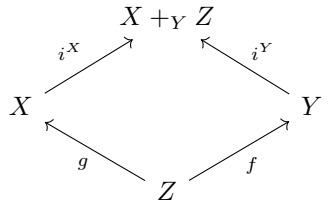
Proof. This is a consequence of Proposition 2.4 of [19] after noting that RMat is the category of R-graphs, the generating data for R-enriched categories. For concreteness and practicality, we offer an explicit construction of pushouts and coproducts here. Let



be a diagram in RMat with



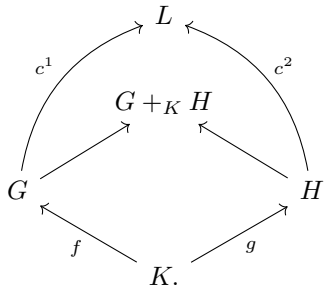
as the underlying diagram of sets. To compute the pushout $G +_K H$ first we take the pushout of sets



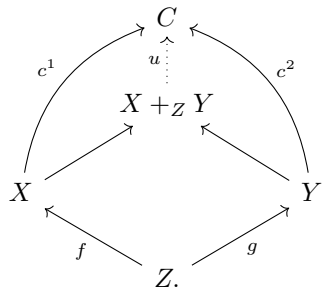
push them forward to get matrices $i^X_*(G)$ and $i^Y_*(H)$ and join them together to get

$$G +_Y H \colon (X +_Y Z) \times (X +_Y Z) \to R = i^X_*(G) + i^Y_*(H)$$

This does indeed define a pushout in RMat. Suppose we have a commutative diagram of R-matrices as follows:



then the underlying diagram of sets induces a unique function u



commuting suitable with c^1 and c^2. The map u is certainly unique, it remains to show that it is well-defined i.e. it satisfies the inequality

$$u_*(G +_K H) \leq L$$

Indeed, for $(x, y) \in C \times C$,

$$\begin{aligned} u_*(G +_K H)(x,y) &= \sum_{(a,b)\in(u\times u)^{-1}(x,y)} G +_K H(a,b) \\ &= \sum_{(a,b)\in(u\times u)^{-1}(x,y)} i^X_*(G)(a,b) + i^Y_*(H)(a,b) \\ &= \sum_{(a,b)\in(u\times u)^{-1}(x,y)} i^X_*(G)(a,b) + \sum_{(a,b)\in(u\times u)^{-1}(x,y)} i^Y_*(H)(a,b) \\ &= u_*(i^X_*(G))(x,y) + u_*(i^Y_*(H))(x,y) \end{aligned}$$

However, because

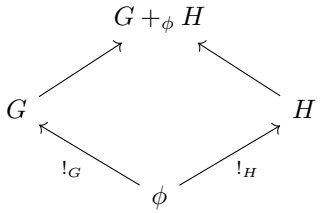
$$u_*(i^X(G)) = c^1_*(G) \text{ and } u_*(i^Y(H)) = c^2_*(G)$$

the above expression is equal to

$$c^1_*(G)(x,y) + c^2_*(H)(x,y)$$

which is less than or equal to $L(x, y)$ because each term is and $+$ is the least upper bound.

For R-matrices $G\colon X \times X \to R$ and $H\colon Y \times Y \to R$, their coproduct is given by the pushout



where ϕ is the unique R-matrix on the empty set and $!_G$ and $!_H$ are the unique morphisms into G and H respectively. ◀

▶ **Proposition 34.** *There is a lax double functor*

$$\blacksquare\colon \mathsf{Open}(\mathsf{RCat}) \to \mathsf{Mat}_R$$

which

- *is the identity on objects,*
- *sends an open R-category $M\colon X \to Y$ to its blackbox $\blacksquare(M)$,*
- *and sends a vertical 2-cell below left*

$$\begin{array}{ccccc} 1_X & \longrightarrow & M & \longleftarrow & 1_Y \\ {\scriptstyle 1_f}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle 1_h} \\ 1_{X'} & \longrightarrow & N & \longleftarrow & 1_{Y'} \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{\blacksquare(M)} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[\blacksquare(N)]{} & Y' \end{array}$$

to the vertical 2-cell above right.

Proof. First observe that this lax double functor is well-defined on 2-cells. This amounts to showing that the inequality

$$\sum_{x \in f^{-1}(x'),\, y \in h^{-1}(y')} M(i(x), j(y)) \leq N(i'(x'), j'(y')) \tag{4}$$

holds. Because g is a morphism of R-matrices, we have that

$$\sum_{a \in g^{-1}(i'(x')),\, b \in g^{-1}(j'(y'))} M(a, b) \leq N(i'(x'), j'(y')) \tag{5}$$

Let $M(i(x), j(y))$ be a term on the left hand side of inequality (4). Then by definition, $x' = f(x)$ and $y' = h(y)$ so $a \in g^{-1}(i'(f(x))$ and $b \in g^{-1}(j'(h(y))$. However, because we started with a 2-cell in $\mathsf{Open}(\mathsf{RCat})$, $i' \circ f = g \circ i$ and $j' \circ h = g \circ j$ so we can rewrite inequality (5) as

$$\sum_{a \in g^{-1}(g \circ i(x)),\, b \in g^{-1}(g \circ j(y))} M(a, b) \leq N(i'(x'), j'(y'))$$

The term $M(i(x), j(y))$ of the left hand side of inequality (4) is also a term of the left hand side of inequality (5) so we have that

$$M(i(x), i(y)) \leq \sum_{a \in g^{-1}(g \circ i(x)),\, b \in g^{-1}(g \circ j(y))} M(a, b) \leq N(i'(x'), j'(y'))$$

Because each term on the left hand side of (4) is less than the desired quantity, the join of all the terms will be as well. Therefore the lax double functor is well-defined on 2-cells. Note that Mat_R is locally posetal i.e. for every square below left

$$\begin{array}{ccc} X & \xrightarrow{M} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ X' & \xrightarrow[N]{} & Y' \end{array} \qquad \begin{array}{ccc} & 1_X & \\ 1_X & & 1_X \end{array}$$

there is at most one 2-cell filling it. This property makes it so many of the axioms in the definition of lax double functor are satisfied trivially. It suffices to show that the globular composition and identity comparisons exist. The identity morphism in $\mathsf{Open}(\mathsf{RCat})$ on a set X is the cospan above right. The blackbox of this cospan is equal to the identity matrix on X, so the identity comparison is the identity. The composition comparison $\blacksquare(M)\blacksquare(N) \leq \blacksquare(M \circ N)$ follows from the chain of inequalities

$$\begin{aligned} \blacksquare(M)\blacksquare(N) &= \sum_{y \in Y} \blacksquare(M)(x, y)\blacksquare(N)(y, z) \\ &= \sum_{y \in Y} M(i(x), j(y))N(i'(y), j'(z)) \\ &= (M +_{1_Y} N)^2 \\ &\leq \sum_{n \geq 0} (M +_{1_Y} N)^n (i(x), j'(z)) \\ &= \blacksquare(M \circ N)(x, z) \end{aligned}$$

Therefore, $\blacksquare$ is a lax double functor. ◀

Preorder-Constrained Simulation for Nondeterministic Automata

Koko Muroya
RIMS, Kyoto University, Japan

Takahiro Sanada
RIMS, Kyoto University, Japan

Natsuki Urabe
National Institute of Informatics, Tokyo, Japan

Abstract

We describe our ongoing work on generalizing some quantitatively constrained notions of weak simulation up-to that are recently introduced for deterministic systems modeling program execution. We present and discuss a new notion dubbed *preorder-constrained simulation* that allows comparison between words using a preorder, instead of equality.

2012 ACM Subject Classification Theory of computation → Verification by model checking

Keywords and phrases simulation, weak simulation, up-to technique, language inclusion, preorder

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.21

Category Early Ideas

Funding The first and second authors are supported by JST, ACT-X Grant No. JPMJAX190U, Japan. The third author is supported by JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603).

1 Introduction: Simulation Notions with Bounded Number of Steps

In the literature of program semantics, coinductive techniques have often been used to establish equivalence between program behaviors. A recent approach utilizes weak simulations with quantitative constraints on the length of terminating runs. These constraints enable comparison of execution cost for programs, in terms of the number of execution steps it takes for a program to terminate.

One example is Accattoli et al.'s notion called *improvement* [1]. It was used to show that certain rewriting of a program before execution not only preserves the execution result, but also *improves* the execution cost by requiring less execution steps. Another example was used in the first author's previous work [9]. It is dubbed (Q, Q_1, Q_2)-*simulation*, parameterized by a triple (Q, Q_1, Q_2) of preorders on natural numbers, i.e. on lengths of runs. The first preorder Q is used to compare lengths of accepted runs, and it generalizes the "greater-than-or-equal" preorder $\geq$ used by improvements. The other two preorders Q_1, Q_2 are for additionally incorporating the so-called *up-to* technique. Subtle conditions on these preorders are identified in *loc. cit.* to make the combination of weak simulations and the up-to technique work.

These two notions are both designed for unlabeled deterministic transition systems, which can model execution of deterministic programs only. We aim to pursue the idea of constraining terminating, or accepted, runs, in a more general setting. This abstract describes our ongoing work on generalizing (Q, Q_1, Q_2)-simulations to nondeterministic automata. We present a novel notion of *preorder-constrained simulation* that is a weak simulation up-to constrained by preorders on words, not on natural numbers. It entails a generalized notion of language inclusion that compares words using a preorder instead of equality.

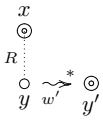
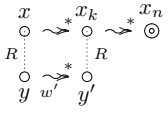


Figure 1 Def. 3 (**Final** and **Step**).



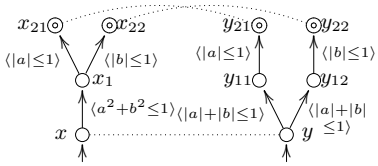


Figure 2 Ex. 4.

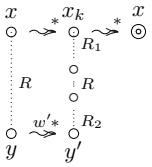


Figure 3 Def. 9.

2 Main Contribution

Let $A_k = (X_k, \Sigma, \rightsquigarrow_k \subseteq X_k \times \Sigma \times X_k, F_k \subseteq X_k)$ $(k \in \{1,2\})$ be nondeterministic automata, $x \in X_1$ and $y \in X_2$, and $L^*_{A_1}(x), L^*_{A_2}(y) \subseteq \Sigma^*$ be the set of words accepted from x and y respectively. The ordinary simulation notion [7] proves language inclusion $L^*_{A_1}(x) \subseteq L^*_{A_2}(y)$. Instead, our simulation notion proves $\mathcal{Q}$-*trace inclusion*.

▶ **Def. 1.** For a preorder $\mathcal{Q} \subseteq \Sigma^* \times \Sigma^*$, we write $x \preceq_{\mathcal{Q}} y$ and say $\mathcal{Q}$-trace inclusion holds between x and y when $\forall w \in L^*_{A_1}(x).\ \exists w' \in L^*_{A_2}(y).\ w \mathcal{Q} w'$.

▶ **Example 2.**

i) when $\mathcal{Q}$ is the equality, $x \preceq_{\mathcal{Q}} y$ iff $L^*_{A_1}(x) \subseteq L^*_{A_2}(y)$.

ii) When Σ contains a special letter τ, and $w \mathcal{Q} w'$ means that w and w' are the same except for τ, then $x \preceq_{\mathcal{Q}} y$ iff *weak language inclusion*, i.e. language inclusion ignoring τ, holds.

iii) When $w \mathcal{Q} w'$ means that w is a subword of w', $x \preceq_{\mathcal{Q}} y$ iff for each $w \in L^*_{A_1}(x)$ there exists $w' \in L^*_{A_2}(y)$ such that w is a subword of w'.

iv) When Σ is the powerset 2^{AP} of some set AP and $a_1 \dots a_k \mathcal{Q} a'_1 \dots a'_{k'}$ means that $k = k'$ and $a_i \subseteq a'_i$ for each $i \in \{1, \dots, k\}$, then $x \preceq_{\mathcal{Q}} y$ iff for each $a_1 \dots a_k \in L^*_{A_1}(x)$ there exists $a'_1 \dots a'_k \in L^*_{A_2}(y)$ such that $a_i \subseteq a'_i$ for each $i \in \{1, \dots, k\}$.

2.1 Preorder-Constrained Simulation without up-to

We hereby introduce a new simulation notion for witnessing $\mathcal{Q}$-trace inclusion.

▶ **Def. 3.** We call $R \subseteq X_1 \times X_2$ a $\mathcal{Q}$-*constrained simulation* from A_1 to A_2 if, for any $(x, y) \in R$, the following holds (see also Fig. 1).

Final: If $x \in F_1$ then there exist $w' \in \Sigma^*$ and $y' \in F_2$ such that $\varepsilon \mathcal{Q} w'$ and $y \overset{w'}{\rightsquigarrow}{}^*_2\ y'$.

Step: For each $a_1 \dots a_n \in \Sigma^+$ and $x_1 \dots x_n \in X_1^+$ such that $x \overset{a_1}{\rightsquigarrow} x_1 \overset{a_2}{\rightsquigarrow} \cdots \overset{a_n}{\rightsquigarrow} x_n$ and $x_n \in F_1$, there exist $k \in \{1, \dots, n\}$, $w' \in \Sigma^*$ and $y' \in X_2$ such that $a_1, \dots, a_k \mathcal{Q} w'$, $y \overset{w'}{\rightsquigarrow}{}^*_2\ y'$ and i) $x_k R y'$ or ii) $k = n$ and $y' \in F_2$.

▶ **Example 4.** We continue Ex. 2(iv). Let $\mathsf{AP} := \mathbb{R}^2$. For a formula $\varphi(a, b)$ with free variables a and b, let $\langle \varphi \rangle := \{(a, b) \in \mathbb{R}^2 \mid \varphi(a, b)\}$. For nondeterministic automata illustrated in Fig. 2, as $\langle a^2 + b^2 \le 1 \rangle \subseteq \langle |a| + |b| \le 1 \rangle$, $R := \{(x, y), (x_{21}, y_{21}), (x_{22}, y_{22})\}$ is a $\mathcal{Q}$-constrained simulation. Note that x_1, y_{11} and y_{12} are not involved by R.

▶ **Prop. 5** (soundness). *If $\mathcal{Q}$ is closed under concatenation (i.e. $w_1 \mathcal{Q} w'_1$ and $w'_2 \mathcal{Q} w'_2$ imply $w_1 w_2 Q w'_1 w'_2$), xRy implies $x \preceq_{\mathcal{Q}} y$.* ◀

Unfortunately, it seems that $\mathcal{Q}$-constrained simulation is not practicable as it is hard to check if given R satisfies **Step** in Def. 3 for all $a_1 \dots a_n \in \Sigma^+$ and $x_1 \dots x_n \in X^+$. In fact, by letting $R := \{(x, y) \mid x \preceq_{\mathcal{Q}} y\}$, we can easily see that $\preceq_{\mathcal{Q}}$ is a $\mathcal{Q}$-constrained simulation.

▶ **Prop. 6** (completeness). *If $\mathcal{Q}$ is closed under concatenation, there exists a $\mathcal{Q}$-constrained simulation R such that $x \preceq_{\mathcal{Q}} y$ implies xRy.* ◀

Position	Player	Moves	
(w, x, y) $\in \Sigma^* \times X_1 \times X_2$	Challenger	(wa, x', y) s.t. $x \overset{a}{\rightsquigarrow}_1 x'$	choose a successor state and enqueue the label
		$(\checkmark, w, x, y)$ when $x \in F_1$	declare the last turn
(w, x', y) $\in \Sigma^+ \times X_1 \times X_2$	Simulator	(w, x', y)	skip the turn
		(ε, x', y') s.t. $\exists w' \in \Sigma^*.\ y \overset{w'*}{\rightsquigarrow}_2 y'$ and $w \mathcal{Q} w'$	dequeue all, and simulate it
$(\checkmark, w, x, y)$ $\in \{\checkmark\} \times \Sigma^+ \times X_1 \times X_2$		sim-win if $\exists w' \in \Sigma^*.\ y \overset{w'*}{\rightsquigarrow}_2 y'$, $w \mathcal{Q} w'$ and $y' \in F_2$	dequeue all and simulate it so that accepting state is reached

Figure 4 Two-player game characterizing $\mathcal{Q}$-constrained simulation. Simulator wins if sim-win is reached, Challenger gets stuck, or a play continues infinitely.

This means that existence of a $\mathcal{Q}$-constrained simulation relating two states is very difficult to determine in many cases. For example, ordinary language inclusion between nondeterministic automata (i.e. $\mathcal{Q}$-trace inclusion when $\mathcal{Q}$ is the equality) is known to be PSPACE-complete [8]. We therefore consider approximating $\mathcal{Q}$-constrained simulation. We consider fixing $M \in \mathbb{N}$ and replacing **Step** with the following (here $A^{[m,M]} := \bigcup_{m \le i \le M} A^i$):

Step$^{\le M}$**:** For each $a_1 \dots a_n \in \Sigma^{[1,M]}$ and $x_1 \dots x_n \in X_1^{[1,M]}$ such that $x \overset{a_1}{\rightsquigarrow} x_1 \overset{a_2}{\rightsquigarrow} \cdots \overset{a_n}{\rightsquigarrow} x_n$ and either $x_n \in F_1$ or $n = M$, there exist $k \in \{1, \dots, n\}$, $w' \in \Sigma^*$ and $y' \in X_2$ such that $a_1, \dots, a_k \mathcal{Q} w'$, $y \overset{w'*}{\rightsquigarrow}_2 y'$ and i) $x_k R y'$ or ii) $k = n < M$ and $y' \in F_2$.

▶ **Prop. 7.** *Given $x \in X_1$ and $y \in X_2$, existence of R satisfying* ***Final*** *,* ***Step***$^{\le M}$ *and xRy can be checked in polynomial time to $|\Sigma|$, $|X_1|$, $|X_2|$ and $\mathcal{T}_M(|\Sigma|, |X_2|))$. Here $\mathcal{T}_M(p, q)$ is the computation time for the following problem: given $w \in \Sigma^{[1,M]}$ and a nondeterministic automaton whose alphabet and state space have sizes of p and q, check if the set $\{w' \mid w \mathcal{Q} w'\}$ intersects with the language of the automaton.* ◀

As **Step**$^{\le M}$ implies **Step**, Prop. 5 still holds after the modification. Moreover, by Prop. 7, if M is fixed and $\mathcal{T}_M(|\Sigma|, |X_2|)$ is polynomial to $|X_2|$ and $|\Sigma|$ (it holds for all $\mathcal{Q}$ illustrated above), then existence of a $\mathcal{Q}$-constrained simulation relating two states can be checked in polynomial time.

We conclude this section by giving a game theoretic characterization for $\mathcal{Q}$-constrained simulations, namely the safety game in Fig. 4 played by Challenger and Simulator.

▶ **Prop. 8.** *Simulator is winning in the two-player game in Fig. 4 from a state (ε, x, y) if and only if there exists a $\mathcal{Q}$-constrained simulation R such that xRy.* ◀

Intuitively, $w \in \Sigma^*$ in Fig. 4 is understood as a *queue* that saves labels executed on A_1 by Challenger. Basically, Simulator can skip the turn until she can construct a path labeled by a word $w' \in \Sigma^*$ such that $w \mathcal{Q} w'$. However, when an accepting state is reached on A_1, Challenger can also declare the last turn and force Simulator to construct a path immediately, although if Simulator succeeded in constructing a path then Challenger loses.

The modification of $\mathcal{Q}$-constrained simulation stated above, i.e. replacing **Step** with **Step**$^{\le M}$, corresponds to replacing Σ^* and Σ^+ with $\Sigma^{[0,M]}$ and $\Sigma^{[1,M]}$ respectively, and prohibiting Simulator from skipping the turn when $|w| = M$ in Fig. 4.

2.2 Preorder-Constrained Simulation with up-to

We can think of an up-to variant of $\mathcal{Q}$-constrained simulations from A_1 to A_2.

▶ **Def. 9.** Let $R_1 \subseteq X_1 \times X_1$ and $R_2 \subseteq X_2 \times X_2$. A $\mathcal{Q}$-constrained simulation R *up-to* (R_1, R_2) is defined in almost the same manner as Def. 3, except that $x_k R y'$ at the end of **Step** is replaced by $x_k R_1 R R_2 y'$ (see also Fig. 3).

We are in particular interested in $R_1 \subseteq \preceq_{\mathcal{Q}_1}$ and $R_2 \subseteq \preceq_{\mathcal{Q}_2}$. Naturally, R_1 and R_2 cannot be arbitrary to verify soundness $xRy \implies x \preceq_{\mathcal{Q}} y$. They have to be *compatible* with $\preceq_{\mathcal{Q}}$. We should also be aware of that a naïve combination of weak simulations and up-to techniques is known to be unsound, and requires special care [10, 11]. Ex. 2(ii) suggests that $\mathcal{Q}$-constrained simulation is a variant of weak simulation. It turns out that restrictions can be simply on the preorders $\mathcal{Q}, \mathcal{Q}_1, \mathcal{Q}_2$.

▶ **Prop. 10.** *Assume $\mathcal{Q}$ is closed under concatenation. If $R_1 \subseteq \preceq_{\mathcal{Q}_1}$ and $R_2 \subseteq \preceq_{\mathcal{Q}_2}$ for preorders $\mathcal{Q}_1, \mathcal{Q}_2 \subseteq \Sigma^* \times \Sigma^*$ satisfying the following conditions, then xRy implies $x \preceq_{\mathcal{Q}} y$: i) $\mathcal{Q}_1 \mathcal{Q} \mathcal{Q}_2 \subseteq \mathcal{Q}$; and ii) $w \mathcal{Q}_1 w'$ implies $|w| \geq |w'|$.* ◀

Cond. (i) ensures compatibility of $\preceq_{\mathcal{Q}_1}, \preceq_{\mathcal{Q}_2}$ with $\preceq_{\mathcal{Q}}$, and Cond. (ii) ensures safe integration of the up-to technique. They are strongly inspired by (Q, Q_1, Q_2)-simulation for unlabeled and deterministic automata [9].

3 Related Work

The above notion is similar to *buffered simulation* [3], which was developed to enable more relations to witness language inclusion. Buffered simulations allow Simulator to skip his turn, to buffer Challenger's moves and to simulate them later together, which has a similar flavor to our simulation notion (cf. Ex. 4). Hence our simulation notion can be also thought of as a generalization of buffered simulation.

Preorder-constrained simulations allow a quantitative reasoning such as comparing lengths of accepted runs. There exist quantitative simulation notions for comparing costs of weighted automata. Many of them are for probabilistic systems [6, 5, 4]. One simulation notion for automata weighted with costs was introduced as a matrix over real numbers [12]. A methodology for comparing infinite runs of weighted automata is also known [2]. In contrast to weighted automata, which are labeled with both letters and weights, our target is automata labeled with letters only. Quantities appear in the set of words, in our approach.

4 Research Directions

Our simulation notion focuses on finite languages. As is the case for the ordinary simulation notion, our notion may fail to prove inclusion of finite languages when there is no inclusion of infinite languages. We are looking into possible solutions.

We suspect that Cond. (ii) of Prop. 5, whose analogues are also in existing notions of weak simulation up-to, is too strong. We think $\mathcal{Q}_1$ violating Cond. (ii) can be allowed finitely many times. However, at the same time, we should note that the relaxation makes the definition of simulations a global one, which can result in a more complicated algorithm for finding it. We should make sure that it does not ruin efficiency gained by up-to techniques.

Ex. 4 suggests that our simulation notion works well with systems whose alphabet Σ carries an order. Such a system also arises in the study of linear temporal logic (LTL). An LTL formula induces a Büchi automaton labeled with the powerset 2^{AP} of atomic propositions [13]. The alphabet 2^{AP} is ordered by the inclusion, which induces a preorder on $(2^{\mathsf{AP}})^*$.

We are also interested in a categorical study of our simulation notion. One possible strategy would be to use the category **PreOrd** of preordered sets as the base category. The nondeterministic branching would be then captured by the powerset functor (or possibly a monad) $\mathcal{P}$ lifted to **PreOrd**. The categorical generalization might allow us to transfer our simulation notion to systems with other branching types, e.g. probabilistic one.

References

1 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The machinery of interaction. In *PPDP 2020*, pages 4:1–4:15. ACM, 2020.

2 Suguman Bansal, Swarat Chaudhuri, and Moshe Y. Vardi. Comparator automata in quantitative verification. In *FoSSaCS 2018*, volume 10803 of *Lecture Notes in Computer Science*, pages 420–437. Springer, 2018.

3 Milka Hutagalung, Martin Lange, and Étienne Lozes. Buffered simulation games for büchi automata. In Zoltán Ésik and Zoltán Fülöp, editors, *AFL 2014*, volume 151 of *EPTCS*, pages 286–300, 2014.

4 Bart Jacobs and Jesse Hughes. Simulations in coalgebra. *Electronic Notes in Theoretical Computer Science*, 82(1):128–149, 2003.

5 Bengt Jonsson and Kim Guldstrand Larsen. Specification and refinement of probabilistic processes. In *LICS 1991*, pages 266–277. IEEE Computer Society, 1991.

6 Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, 1991.

7 Nancy A. Lynch and Frits W. Vaandrager. Forward and backward simulations: I. Untimed systems. *Inf. Comput.*, 121(2):214–233, 1995. doi:10.1006/inco.1995.1134.

8 A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory (swat 1972)*, pages 125–129, 1972. doi:10.1109/SWAT.1972.29.

9 Koko Muroya. *Hypernet Semantics of Programming Languages*. PhD thesis, University of Birmingham, 2020.

10 Damien Pous. Up-to techniques for weak bisimulation. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 730–741. Springer, 2005.

11 Damien Pous. New up-to techniques for weak bisimulation. *Theoretical Computer Science*, 380(1):164–180, 2007. Automata, Languages and Programming.

12 Natsuki Urabe and Ichiro Hasuo. Generic forward and backward simulations III: quantitative simulations by matrices. In *CONCUR 2014*, volume 8704 of *Lecture Notes in Computer Science*, pages 451–466. Springer, 2014.

13 Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

Quantitative Polynomial Functors

Georgi Nakov
Department of Computer and Information Sciences, University of Strathclyde, UK

Fredrik Nordvall Forsberg
Department of Computer and Information Sciences, University of Strathclyde, UK

Abstract

We investigate containers and polynomial functors in Quantitative Type Theory, and give initial algebra semantics of inductive data types in the presence of linearity. We show that reasoning by induction is supported, and equivalent to initiality, also in the linear setting.

2012 ACM Subject Classification Theory of computation → Type theory; Theory of computation → Linear logic

Keywords and phrases quantitative type theory, polynomial functors, inductive data types

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.22

Category Early Ideas

1 Quantitative Type Theory

Quantitative Type Theory (QTT) [3, 12] combines linear and dependent types, allowing for tracking and reasoning about resource usage of programs. Such a combination is non-trivial as there is no obvious answer how to treat terms occurring in type formation. Previous attempts include the Linear Logical Framework [6], and the work of Krishnaswami, Pradic and Benton [11] and Vákár [14], based on Linear/Non-Linear logic [5], in which the context is split into intuitionistic and linear parts, and each type is allowed to depend on only one of the two. QTT differs by maintaining a single context, where each variable is annotated with resource information (for a similar approach, see Orchard et al. [13], Fu et al. [8] and Abel and Bernardy [2]). For example, consider the judgement (where $\mathsf{Fin} : \mathbb{N} \to \mathsf{Type}$ is a type family with $\mathsf{Fin}(n)$ consisting of natural numbers smaller than n):

$$n \overset{0}{:} \mathbb{N}, x \overset{2}{:} \mathsf{Fin}(n) \vdash x + x \overset{1}{:} \mathsf{Fin}(2n)$$

The variables on the left of the turnstile form the context of the judgement. Each one is annotated with a quantity, taken from a fixed semiring R (here, $R = (\mathbb{N}, +, *)$). These quantities denote how many the times the variables must be used in the term on the right. Hence here we see that n is used 0 times, since it only occurs in types, and x is used twice. A context can be pointwise scaled by an element $\pi \in R$, and two contexts with the same "underlying non-resource-annotated context" can be added pointwise:

$$\pi(\Gamma, x \overset{\rho}{:} S) = \pi\Gamma, x \overset{\pi\rho}{:} S$$
$$(\Gamma_1, x \overset{\rho_1}{:} S) + (\Gamma_2, x \overset{\rho_2}{:} S) = (\Gamma_1 + \Gamma_2), x \overset{\rho_1+\rho_2}{:} S, \text{ if } 0\Gamma_1 = 0\Gamma_2.$$

While arbitrary elements from R may occur as annotations in the context, only the 0 and 1 are valid annotations for the conclusion, in order to retain admissibility of substitution [3, § 2.3]. Such a restriction effectively splits the theory in two fragments. Terms in the so-called

$$\frac{0\Gamma \vdash X : \mathsf{Type} \quad 0\Gamma, x \overset{0}{:} X \vdash Y : \mathsf{Type}}{0\Gamma \vdash (x \overset{\rho}{:} X) \to Y : \mathsf{Type}} \qquad \frac{\Gamma, x \overset{\rho}{:} X \vdash t : Y}{\Gamma \vdash \lambda x.\ t : (x \overset{\rho}{:} X) \to Y}$$

$$\frac{\Gamma_1 \vdash f : (x \overset{\rho}{:} X) \to Y \quad \Gamma_2 \vdash t : S \quad 0\Gamma_1 = 0\Gamma_2}{\Gamma_1 + \pi\Gamma_2 \vdash f(t) : Y[t/x]}$$

$$\frac{0\Gamma \vdash X : \mathsf{Type} \quad 0\Gamma, x \overset{0}{:} X \vdash Y : \mathsf{Type}}{0\Gamma \vdash (x \overset{\rho}{:} X) \otimes Y : \mathsf{Type}} \qquad \frac{\Gamma_1 \vdash s : X \quad \Gamma_2 \vdash t : Y[s/x] \quad 0\Gamma_1 = 0\Gamma_2}{\rho\Gamma_1 + \Gamma_2 \vdash (s,t) : (x \overset{\rho}{:} X) \otimes Y}$$

$$\frac{0\Gamma_1, z \overset{0}{:} (x \overset{\rho}{:} X) \otimes Y \vdash U : \mathsf{Type} \quad \Gamma_1 \vdash w : (x \overset{\rho}{:} X) \otimes Y \quad \Gamma_2, x \overset{\rho}{:} X, y \overset{1}{:} Y \vdash u : U[(x,y)/z] \quad 0\Gamma_1 = 0\Gamma_2}{\Gamma_1 + \Gamma_2 \vdash \ \text{let } (x,y) = s \text{ in } t : U[w/z]}$$

Figure 1 Typing rules for the dependent function and dependent tensor types.

0-fragment bear no computational content, while the inhabitants of the 1-fragment are computationally relevant[1]. The core insight is that contemplating variables in types is always possible, even for already consumed ones, and thus type formation happens in the 0-fragment.

The typing rules in the system now carry resource tracking duty, which is especially evident in types with binders – see Figure 1. Function type now records how many times its arguments are used; e.g. a function $f : (x \overset{\rho}{:} X) \to Y$ must be supplied with ρ many copies of x. (If the type family Y does not depend on X, we will use the simplified notation $f : X \xrightarrow{\rho} Y$ for $f : (x \overset{\rho}{:} X) \to Y$.) In a similar vein, the second component of the dependent pair $t : (x \overset{\rho}{:} A) \otimes B$ requires ρ many copies of the first component.

2 Quantitative Containers

We want to extend QTT with data types to program with and reason about, but an ad-hoc approach of writing out introduction and elimination rules for each data type is cumbersome and error-prone. A principled solution to adding inductive data types in a traditional setting is provided by the theory of polynomial functors [9] and containers [1]. We build the syntactic category of closed types and linear functions in QTT and define polynomial functors on it. We can then systematically derive appropriate elimination rules for their initial algebras.

2.1 Quantitative container functors on the category of (closed) types and linear functions

In traditional type theory, a container functor is a functor on types of the form $F_{S,P}(X) = (s : S) \times (P[s] \to X)$, where $(x : A) \times B[x]$ is the dependent pair type. We think of the parameter S as a type of *shapes*, and the family $P : S \to \mathsf{Type}$ as a type of *positions* for each shape. An element of $F_{S,P}(X)$ is describing how to fill the container with "payloads" from X: a choice of a shape $s : S$, and an element of X for every position in $P[s]$. The following is the obvious transfer of this idea to a quantitative setting:

[1] From now on, we omit the annotation on terms in the 1-fragment on the right of the turnstile in typing judgements – that is, $\Gamma \vdash t : A$ tacitly means $\Gamma \vdash t \overset{1}{:} A$.

▶ **Proposition 1.** *Let $\mathcal{C}$ be the category of closed types and linear functions: its objects are types $\vdash X$, and morphisms are functions $\vdash f : X \xrightarrow{1} Y$. For fixed $S : \mathsf{Type}$ and $P : S \xrightarrow{0} \mathsf{Type}$, the mapping $F_{S,P}(X) = (s \overset{1}{:} S) \otimes (P[s] \xrightarrow{1} X)$ is a functor $\mathcal{C} \to \mathcal{C}$.*

We call any functor isomorphic to one of the form $F_{S,P}$ a *quantitative container*. The above proposition can be generalised to types and functions over an arbitrary, fixed context of the shape $\Gamma = 0\Gamma$, i.e. where all variables are annotated with 0.

2.2 Induction principles and initial algebras

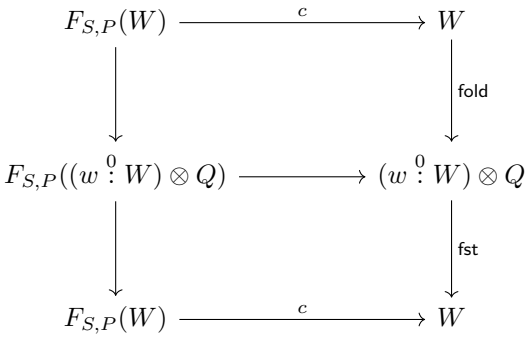
Recall that an F-algebra for an endofunctor $F : \mathcal{C} \to \mathcal{C}$ is a pair $(A, a : F(A) \to A)$, where A is an object of $\mathcal{C}$ and a is a $\mathcal{C}$-morphism. A morphism between F-algebras (A, a) and (B, b) is a map $f : A \to B$ in $\mathcal{C}$, such that $f \circ a = b \circ F(f)$. Inductive data types are initial algebras: the algebra map corresponds to the introduction rule of the type, and the mediating map corresponds to the elimination rule. A priori, this only gives *non-dependent* elimination rules (also known as recursion principles), but by exploiting the uniqueness of the mediating map, we can also derive *dependent* elimination rules (induction principles), and vice versa:

▶ **Theorem 2.** *Let $\mathbf{W} := (W, c : F_{S,P}(W) \xrightarrow{1} W)$ be an $F_{S,P}$-algebra. $\mathbf{W}$ is initial if and only if the following induction principle holds:*

$$\frac{w \overset{0}{:} W \vdash Q : \mathsf{Type} \quad \vdash M : (s \overset{1}{:} S) \to (h \overset{0}{:} P[s] \xrightarrow{1} W) \to ((p \overset{1}{:} P[s]) \to Q(h(p))) \xrightarrow{1} Q(c(s,h))}{\vdash \mathsf{elim}(Q, M) : (x \overset{1}{:} W) \to Q[x]}$$

Proof (sketch). The proof is based on the standard construction (see e.g. Awodey, Gambino and Sojakova [4] or Hermida and Jacobs [9]), but accounting for linearity.

Assuming that $\mathbf{W}$ is initial and the premises of the elimination rule, we build an $F_{S,P}$-algebra on the dependent tensor type $(w \overset{0}{:} W) \otimes Q$, and get a unique mediating morphism $\mathsf{fold} : W \xrightarrow{1} (w \overset{0}{:} W) \otimes Q$ by initiality. We compose with the second projection $\mathsf{snd} : (x \overset{1}{:} (w \overset{0}{:} W) \otimes Q) \to Q[\mathsf{fst}(x)]$ to get a map $(x \overset{1}{:} W) \to Q[\mathsf{fst}(\mathsf{fold}(x))]$. Note that the use of second projection is admissible due to the annotation of the first component $w \overset{0}{:} W$ – we are free to dispose of w, since it is used 0 times. To show that $\mathsf{snd} \circ \mathsf{fold}$ has the right type, we need to show that $Q[\mathsf{fst}(\mathsf{fold}(x))] = Q[x]$ for every $x : W$, but as this is a type equality, unrestricted use of terms is permissible. The map $\mathsf{fst} : (w \overset{0}{:} W) \otimes Q \xrightarrow{1} W$ is an $F_{S,P}$-algebra morphism, and thus the composite $\mathsf{fst} \circ \mathsf{fold} : W \xrightarrow{1} W$ is also one:



Thus $\mathsf{fst} \circ \mathsf{fold} = \mathsf{id}$ holds by uniqueness of the mediating morphism out of W.

The converse direction follows analogously. ◀

3 Quantitative polynomial functors

However, there is a caveat – most quantitative versions of standard data types are not quantitative containers in the above sense. Consider, for example, the natural numbers, the initial algebra of the polynomial functor $F(X) = \mathbf{1} + X$, or binary trees, the initial algebra of $G(X) = A + X \times X$. Their representations in "container normal form" crucially depend on isomorphisms $(\mathbf{0} \to X) \cong \mathbf{1}$ and $(\mathbf{Bool} \to X) \cong X \times X$, respectively, but their QTT counterparts do not hold: $(\mathbf{0} \xrightarrow{1} X) \ncong \mathbf{I}$ (where $\mathbf{I}$ is the monoidal unit) and $(\mathbf{Bool} \xrightarrow{1} X) \ncong X \otimes X$. Thus we resort to generating the class of *quantitative polynomial functors* inductively by the following grammar:

$$F, G ::= \mathsf{Id} \mid \mathsf{Const}_A \mid F \underline{\otimes} G \mid F \underline{\oplus} G \mid F \,\underline{\&}\, G \mid A \underline{\xrightarrow{1}} X \tag{1}$$

Theorem 2 still holds for this class of functors, with the induction principle reformulated using an appropriately defined predicate lifting $\widehat{F}_X : (Q : X \to \mathsf{Type}) \to (F(X) \to \mathsf{Type})$ for the type of induction hypothesis:

▶ **Theorem 3.** *Let F be a quantitative polynomial functor and $\mathbf{W} := (W, c : F(W) \xrightarrow{1} W)$ an F-algebra. $\mathbf{W}$ is initial if and only if the following induction principle holds:*

$$\frac{w \overset{0}{:} W \vdash Q : \mathsf{Type} \quad \vdash M : ((w \overset{0}{:} F(W)) \otimes \widehat{F}_W(Q, w)) \xrightarrow{1} Q(c(w))}{\vdash \mathsf{elim}(Q, M) : (x \overset{1}{:} W) \to Q[x]}$$

Our proof is a variation on the proof of Theorem 2, using a distributive lemma for the dependent tensor and the predicate lifting:

▶ **Lemma 4.** *For a quantitative polynomial functor F, we have*

$$F((w \overset{0}{:} W) \otimes Q) \cong (w' \overset{0}{:} F(W)) \otimes \widehat{F}_W(Q, w').$$

4 Existence of initial algebras

Categorical models of QTT build upon Categories with families (CwF) [7], a standard framework for models of dependent type theories. A Quantitative CwF [3] consists of an ordinary CwF $\mathcal{C}$, a category of resourced contexts and substitutions $\mathcal{L}$, and data to interpret resourced counterparts of context extensions and terms. A faithful functor $U : \mathcal{L} \to \mathcal{C}$ relates the computationally relevant 1-fragment in $\mathcal{L}$ to the 0-fragment in $\mathcal{C}$.

A concrete model is given by choosing $\mathcal{C} = \mathcal{S}et$ and $\mathcal{L}$ to be the category of assemblies [15] using linear realisability [10]. We can prove existence of initial algebras of finitary quantitative polynomial functors (i.e., generated without $A \underline{\xrightarrow{1}} X$) in the model by reusing the initial algebras in $\mathcal{S}et$, using their universal properties to construct the realisers. Unfortunately, this method does not extend to non-finitary polynomial functors (i.e. those generated using $A \underline{\xrightarrow{1}} X$ in the grammar (1)), since the case for the type constructor $A \underline{\xrightarrow{1}} X$ gives a realiser $r(a)$ for every $a : A$, but not a realisable function. In the future, we hope to overcome this shortcoming.

References

1 Michael Abbott, Thorsten Altenkirch, and Neil Ghani. Categories of containers. *Lecture Notes in Computer Science*, 2620:23–38, 2003. doi:10.1007/3-540-36576-1_2.

2 Andreas Abel and Jean-Philippe Bernardy. A unified view of modalities in type systems. *Proceedings of the ACM on Programming Languages*, 4(ICFP):1–28, 2020. doi:10.1145/3408972.

3 Robert Atkey. Syntax and Semantics of Quantitative Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science - LICS '18*, pages 56–65, New York, New York, USA, 2018. ACM Press. doi:10.1145/3209108.3209189.

4 Steve Awodey, Nicola Gambino, and Kristina Sojakova. Homotopy-initial algebras in type theory. *Journal of the ACM*, 63(6), 2017.

5 P. N. Benton. A mixed linear and non-linear logic: Proofs, terms and models. In *Lecture Notes in Computer Science*, volume 933, pages 121–135. Springer, Berlin, Heidelberg, 1995. doi:10.1007/BFb0022251.

6 Iliano Cervesato and Frank Pfenning. A Linear Logical Framework. *Information and Computation*, 179(1):19–75, 2002. doi:10.1006/inco.2001.2951.

7 Peter Dybjer. Internal type theory. In *Lecture Notes in Computer Science*, volume 1158 LNCS, pages 120–134. Springer, Berlin, Heidelberg, 1996. doi:10.1007/3-540-61780-9_66.

8 Peng Fu, Kohei Kishida, and Peter Selinger. Linear Dependent Type Theory for Quantum Programming Languages: Extended Abstract. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2020: Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 440–453. Association for Computing Machinery, 2020. doi:10.1145/3373718.3394765.

9 Claudio Hermida and Bart Jacobs. Structural Induction and Coinduction in a Fibrational Setting. *Information and Computation*, 145(2):107–152, 1998. doi:10.1006/inco.1998.2725.

10 Naohiko Hoshino. Linear Realizability. In *Computer Science Logic*, volume 4646 LNCS, pages 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-74915-8_32.

11 Neelakantan R. Krishnaswami, Pierre Pradic, and Nick Benton. Integrating Linear and Dependent Types. *ACM SIGPLAN Notices*, 50(1):17–30, 2015. doi:10.1145/2775051.2676969.

12 Conor McBride. I Got Plenty o' Nuttin'. In *Lecture Notes in Computer Science*, volume 9600, pages 207–233. Springer Verlag, 2016. doi:10.1007/978-3-319-30936-1_12.

13 Dominic Orchard, Vilem-Benjamin Liepelt, and Harley Eades III. Quantitative program reasoning with graded modal types. *Proceedings of the ACM on Programming Languages*, 3(ICFP):110:1–110:30, 2019. doi:10.1145/3341714.

14 Matthijs Vákár. *In search of effectful dependent types*. PhD thesis, University of Oxford, 2017.

15 Jaap van Oosten. *Realizability: an introduction to its categorical side*. Elsevier, 2008.

Nawrotzki's Algorithm for the Countable Splitting Lemma, Constructively

Ana Sokolova
Department of Computer Sciences, University of Salzburg, Austria

Harald Woracek
Institute of Analysis and Scientific Computing, TU Wien, Austria

Abstract

We reprove the countable splitting lemma by adapting Nawrotzki's algorithm which produces a sequence that converges to a solution. Our algorithm combines Nawrotzki's approach with taking finite cuts. It is constructive in the sense that each term of the iteratively built approximating sequence as well as the error between the approximants and the solution is computable with finitely many algebraic operations.

2012 ACM Subject Classification Mathematics of computing → Mathematical analysis; Mathematics of computing → Probability and statistics; Theory of computation → Design and analysis of algorithms; Theory of computation → Semantics and reasoning

Keywords and phrases countable splitting lemma, distributions with given marginals, couplings

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.23

Category (Co)algebraic pearls

Acknowledgements We are indebted to Paul Levy for bringing this topic to us by asking us several years ago to figure out some details of Nawrotzki's algorithm, in particular its constructivity.

1 Explanation of what is going on ...

Given a measure μ on a product space $\prod_{i\in I} X_i$, the *j-th marginal* μ_j of μ is the push-forward of μ under the j-th canonical projection $\pi_j\colon \prod_{i\in I} X_i \to X_j$. Explicitly, this is

$$\mu_j(A) := \mu\big(\pi_j^{-1}(A)\big)$$

for all $A \subseteq X_j$ with $\pi_j^{-1}(A)$ being measureable.

In his fundamental paper [21] Strassen investigated the existence of measures on a product $X \times Y$ which have prescribed marginals and satisfy additional constraints of a certain form. The result stated in Theorem 1 below is a corollary of [21, Theorem 11] and known as *Strassen's theorem on stochastic domination*. Curiously, it is not even explicitly stated in Strassen's paper, but only mentioned in one sentence. We state a slightly more general variant taken from [20, Corollary 7][1]. To formulate it, we need some notation.

- Let X be a Hausdorff space, and let $\preccurlyeq$ be a partial order on X which is closed as a subset of $X \times X$. A subset $A \subseteq X$ is *upward closed w.r.t.* $\preccurlyeq$, if

$$\forall x \in X, y \in A.\ y \preccurlyeq x \Rightarrow x \in A.$$

- For two positive Borel measures μ, ν on X we write $\mu \preceq \nu$, if for all upward closed Borel sets $A \subseteq X$ it holds that $\mu(A) \leq \nu(A)$.

[1] A different proof can be found in [16].

▶ **Theorem 1.** *Let X be a Hausdorff space, let $\preccurlyeq$ be a closed partial order on X, and let μ and ν be two probability (Borel-) measures on X. If $\mu \preceq \nu$, then there exists a probability (Borel-) measure Λ on $X \times X$ which has the marginals μ and ν, and whose support is contained in $\preccurlyeq$, i.e. there exists a subset of $\preccurlyeq$ which has Λ-measure 1.*

An important particular case of Theorem 1 is when the base space X is finite or countable with the discrete topology. In the finite case this result is known as the splitting lemma [11, Theorem 4.10], and the latter is what the term "countable splitting lemma" refers to.

Over the years such results were established in different variants and on different levels of generality. For example: Strassen's original theorem [21] is proven for Polish spaces, [14] for completely regular spaces, [20] for Hausdorff spaces, [17] for probability contents instead of measures, [15] for normal measure spaces under finiteness assumptions on $\preccurlyeq$, [6] for measures with values in vector lattices under restrictions on $\preccurlyeq$, [5] for measure spaces where solutions are only required to have the given marginals up to equivalence of measures, [8] for operator valued measures, [9] for products of finitely many Polish spaces and a different proof than Strassen, [13] for Polish spaces adding some further equivalences. Some predecessors of Strassen's work are [15, 19]. A recent line of research where solutions are only required to have the given marginal up to some error is followed in [10] and related papers.

Theorem 1 plays an important role in probability theory and has applications in various areas. For example, it prominently occurs in finance mathematics, e.g. [4, 7], or in computer science, e.g. [3, 1, 2, 10, 11, 12].

The proof of Theorem 1 relies in general on a rather heavy analytic machinery, in particular, on theorems exploiting compactness properties. If X is finite, a required solution Λ can – naturally – be found by an algorithm which terminates after finitely many steps. This fact can be based on various reasoning. For example on elementary manipulations with inequalities, as e.g. in [15, §3], or combinatorial results like the max-flow min-cut theorem or the subforest lemma, as e.g. in [18] or [11, Theorem 4.10].

In the present exposition we deal with the countable discrete case. Our aim is to give a recursive algorithm which produces a sequence $(\Delta_N)_{N\in\mathbb{N}}$ of (discrete) probability measures on $X \times X$ such that

1. each term of the sequence is computable from the inital data μ, ν with a finite number of algebraic operations;
2. the sequence $(\Delta_N)_{N\in\mathbb{N}}$ converges to a solution Λ in the ℓ^1-norm on $X \times X$, in particular it converges pointwise;
3. the speed of pointwise convergence can be controlled in a computable way.

To explain our contribution, it is worthwhile to revisit the presently availabe proofs for the countable discrete case. First, specialising the general proof(s) of Theorem 1 obviously does not lead to an algorithm, since tools like e.g. the Banach-Alaoglu Theorem are used. More interesting are the arguments given in the papers of Kellerer [15, §4] and Nawrotzki [19]. Both are non-constructive, but for different reasons.

- Kellerer's approach is to reduce to the finite cases. Given μ, ν on a countable set, he produces appropriately cut-off data μ_N, ν_N, $N \in \mathbb{N}$, and solves the problem for those. This gives a measure Λ_N on X, which solves the problem up to the index N. Each measure Λ_N can be computed in finitely many steps. Sending the cut-off point N to infinity leads to existence of a solution for the full data μ, ν. The masses of the measures Λ_N may oscillate, and therefore the sequence $(\Lambda_N)_{N\in\mathbb{N}}$ need not be convergent. However, each accumulation point of the sequence $(\Lambda_N)_{N\in\mathbb{N}}$ will be a solution.
  What makes the method non-constructive is that accumulation points *exist by compactness* (in this case applied in the form of the Heine-Borel Theorem).

- Nawrotzki's approach is to produce a sequence $(\Lambda_N)_{N\in\mathbb{N}}$, which does not necessarily solve the problem on any finite section, but still converges to a solution. His construction ensures that the masses of the measures Λ_N are nonincreasing on points of the diagonal and nondecreasing off the diagonal. This ensures that passing to subsequences is not necessary.

  What makes the method non-constructive is that defining the measures Λ_N requires to evaluate *sums of infinite series* and *infima of infinite sets* of real numbers.

Our idea to produce $(\Delta_N)_{N\in\mathbb{N}}$ with **1.**–**3.** above, is to combine the approaches: we apply Nawrotzki's algorithm to appropriately truncated sequences to ensure computability, and control the error which is made by passing to cut-off's to ensure convergence.

2 Nawrotzki's algorithm

In [19], which preceeds the work of Strassen, Nawrotzki proved a discrete version of Strassen's theorem. In our present language his result reads as follows.

▶ **Theorem 2.** *Let $\mu = (\mu_n)_{n\in\mathbb{N}}$ and $\nu = (\nu_n)_{n\in\mathbb{N}}$ be sequences of real numbers, such that*

$$\forall n \in \mathbb{N}.\ \mu_n \geq 0 \wedge \nu_n \geq 0 \quad \text{and} \quad \sum_{n\in\mathbb{N}} \mu_n = \sum_{n\in\mathbb{N}} \nu_n = 1, \tag{1}$$

Moreover, let $\preccurlyeq$ be a partial order on $\mathbb{N}$.

If it holds that

$$\forall R \subseteq \mathbb{N} \text{ upwards closed w.r.t. } \preccurlyeq.\ \sum_{n\in R} \mu_n \leq \sum_{n\in R} \nu_n, \tag{2}$$

then there exists an infinite matrix $\Lambda = (\lambda_{n,m})_{n,m\in\mathbb{N}}$ of real numbers, such that

$$\forall n, m \in \mathbb{N}.\ \lambda_{n,m} \geq 0 \quad \text{and} \quad \sum_{n,m\in\mathbb{N}} \lambda_{n,m} = 1, \tag{3}$$

$$\forall n, m \in \mathbb{N}.\ \lambda_{n,m} \neq 0 \Rightarrow n \preccurlyeq m, \tag{4}$$

$$\forall n \in \mathbb{N}.\ \sum_{m\in\mathbb{N}} \lambda_{n,m} = \mu_n, \tag{5}$$

$$\forall m \in \mathbb{N}.\ \sum_{n\in\mathbb{N}} \lambda_{n,m} = \nu_m. \tag{6}$$

In this section we present Nawrotzki's argument in a structured way including all details. This provides an in-depth understanding of his work, and this is necessary to make appropriate adaption to the algorithm later on (in Section 3).

▶ Remark 3. Before we dive into the formulas and proofs, which are a bit technical and lengthy, let us give an intuition for what is going to happen.

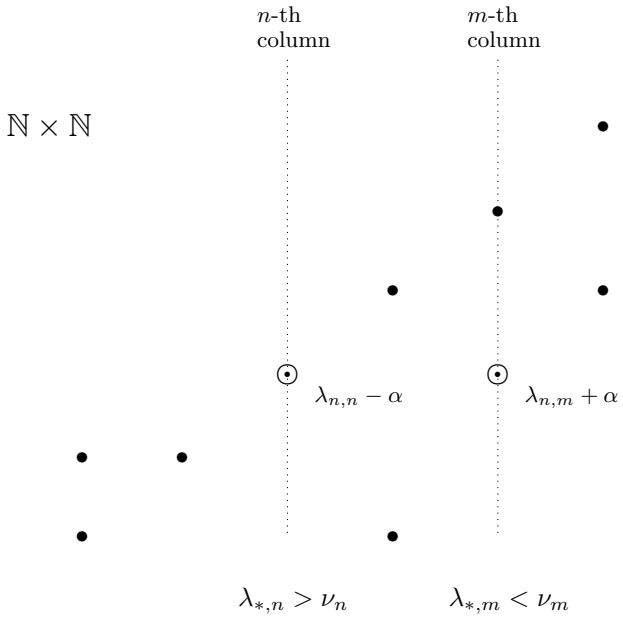
Assume we are given data μ_n, ν_m satisfying Equations (1) and (2) and a (probably bad) approximation of a solution $\lambda_{n,m}$ that satisfies Equations (3) and (4), as well as Equation (5). Note that achieving correctness of one marginal, i.e. satisfying Equation (5), is very easy; for example already the diagonal matrix with μ_n's on the diagonal will satisfy this.

If the column sums do not give the correct results as required by Equation (6), it must be that some of them are larger than the target value and some of them are smaller since the total sum is always 1. Now we want to modify the values $\lambda_{n,m}$ to improve the approximation, i.e., make the error in Equation (6) smaller while retaining all other properties. Most importantly, we have to ensure that Equation (2), also known as *stochastic dominance*, is inherited. In addition, we want to make the modification in such a way that:

1. At each place (n, m) entries change monotonically when repeating the step in the algorithm. This is achieved by having diagonal entries nonincreasing and off-diagonal entries nondecreasing. This will guarantee existence of a limit.
2. Make sure that the pattern of which column sums are too large and which are too small is inherited with exception that some column sums may become correct. This will guarantee that the algorithm can proceed appropriately.

The algorithm proceeds in steps. In each step exactly two values of the matrix change: one at the diagonal at position (n, n) and another in the same row at position (n, m) such that Equation (6) fails for n and m, as pictured below. The new values are $\lambda'_{n,n} = \lambda_{n,n} - \alpha$ and $\lambda'_{n,m} = \lambda_{n,m} + \alpha$, where α is chosen such that still $\lambda'_{*,n} \geq \nu_n$, $\lambda'_{*,m} \leq \nu_m$.

In the picture, filled circles indicate those points where our approximation has nonzero entries, circled dots mark the changes made by one step of the algorithm, and $\alpha > 0$ is the correction term whose exact definition (see Definition 7) is taylor made so that the above explained requirements are met.



The next result, Proposition 5, is the first crucial ingredient to Nawrotzki's algorithm (out of two; the second is Proposition 10 further below). It will ensure that in the limit a solution is obtained. To formulate it, we need additional notation.

▶ **Definition 4.** *Let $\preccurlyeq$ be a partial order on $\mathbb{N}$. For each $(n, m) \in \mathbb{N} \times \mathbb{N}$ with $n \prec m$, we denote*

$$\mathcal{R}_{n,m} := \big\{R \subseteq \mathbb{N} \mid n \notin R, m \in R, R \text{ upward closed w.r.t. } \preccurlyeq\big\}.$$

Note that $\mathcal{R}_{n,m}$ is always nonempty. For example, we have

$$\{l \in \mathbb{N} \mid m \preccurlyeq l\} \in \mathcal{R}_{n,m}.$$

▶ **Proposition 5.** *Assume that μ, ν, and $\preccurlyeq$, satisfy Equation* (1) *and Equation* (2)*. If for each pair $(n,m) \in \mathbb{N} \times \mathbb{N}$ with $n \prec m$ at least one of*

$$\mu_n \leq \nu_n, \tag{7}$$

$$\mu_m \geq \nu_m, \tag{8}$$

$$\inf_{R \in \mathcal{R}_{n,m}} \sum_{l \in R} (\nu_l - \mu_l) = 0, \tag{9}$$

holds, then $\mu = \nu$.

Note here that all series in Equation (9) converge absolutely and that by Equation (2) the infimum in Equation (9) is nonnegative. Moreover, in an algorithm acting as explained in Remark 3 above (and defined in precise mathematical terms in Definition 7 below), using $\mathcal{R}_{n,m}$ instead of all upwards closed sets is sufficient to retain Equation (2). This is because for upwards closed sets which are not in $\mathcal{R}_{n,m}$, Equation (2) is trivially inherited.

In the proof of Proposition 5, we use the following simple fact.

▶ **Lemma 6.** *Assume that μ, ν, and $\preccurlyeq$, satisfy Equation* (1) *and Equation* (2)*. Further, let $R_1, R_2, \dots$ be a (finite or infinite) sequence of upward closed (w.r.t. $\preccurlyeq$) subsets of $\mathbb{N}$, and set*

$$R := \bigcup_k R_k.$$

Then R is upward closed, and

$$\sum_{l \in R} (\nu_l - \mu_l) \leq \sum_k \sum_{l \in R_k} (\nu_l - \mu_l).$$

Proof. Since $|\nu_l - \mu_l| \leq \nu_l + \mu_l$, the series on the left side converges absolutely. Hence, we may rearrange summands without changing its value. Now write R as the disjoint union

$$R = \dot{\bigcup_k} R'_k$$

where

$$R'_k := R_k \setminus \bigcup_{j<k} R_j.$$

Then

$$\sum_{l \in R} (\nu_l - \mu_l) = \sum_k \sum_{l \in R'_k} (\nu_l - \mu_l).$$

For each k we have

$$\sum_{l \in R_k} (\nu_l - \mu_l) = \sum_{l \in R'_k} (\nu_l - \mu_l) + \sum_{R_k \cap \bigcup_{j<k} R_j} (\nu_l - \mu_l).$$

The set $R_k \cap \bigcup_{j<k} R_j$ is upward closed, and hence the second summand on the right side is nonnegative. This shows that

$$\sum_{l \in R'_k} (\nu_l - \mu_l) \leq \sum_{l \in R_k} (\nu_l - \mu_l)$$

for all k. ◀

Proof of Proposition 5. It is enough to show that $\mu_n \leq \nu_n$ for all $n \in \mathbb{N}$. Assume towards a contradiction that there exists $n \in \mathbb{N}$ with $\mu_n > \nu_n$, and fix one with this property. Moreover, choose $\epsilon > 0$ small enough, say,

$$\epsilon := \frac{1}{3}(\mu_n - \nu_n).$$

By the assumption of the proposition we know that for each $m \in \mathbb{N}$ with $m \succ n$ at least one of

- $\mu_m \geq \nu_m$,
- $\inf_{R \in \mathcal{R}_{n,m}} \sum_{l \in R}(\nu_l - \mu_l) = 0$,

must hold.

Consider the set where the second case takes place

$$H := \Big\{ m \in \mathbb{N} \mid n \prec m, \inf_{R \in \mathcal{R}_{n,m}} \sum_{l \in R}(\nu_l - \mu_l) = 0 \Big\}.$$

If $H = \emptyset$, it is easy to reach a contradiction. Namely, if $\mu_m \geq \nu_m$ for all $m \succ n$, then

$$\sum_{m \succcurlyeq n} \mu_m > \sum_{m \succcurlyeq n} \nu_m,$$

and this contradicts Equation (2).

If $H \neq \emptyset$, we argue as follows. For each $m \in H$ choose $R_m \in \mathcal{R}_{n,m}$, such that

$$\sum_{l \in R_m} (\nu_l - \mu_l) \leq \frac{\epsilon}{2^m},$$

and set $R := \bigcup_{m \in H} R_m$. Then $H \subseteq R$, $n \notin R$, and

$$\sum_{l \in R}(\nu_l - \mu_l) \leq \sum_{m \in H} \sum_{l \in R_m} (\nu_l - \mu_l) \leq \sum_{m \in H} \frac{\epsilon}{2^m} \leq 2\epsilon.$$

Consider the upward closed set

$$R' := R \cup \{ l \in \mathbb{N} \mid n \prec l \}.$$

If $l \in R' \setminus R$, then $n \prec l$ and $l \notin H$. Thus we must have $\mu_l \geq \nu_l$. From this we see that

$$0 \leq \sum_{l \in R'}(\nu_l - \mu_l) = \sum_{l \in R}(\nu_l - \mu_l) + \sum_{l \in R' \setminus R} (\nu_l - \mu_l) \leq \sum_{l \in R}(\nu_l - \mu_l) \leq 2\epsilon.$$

The set $R' \cup \{n\}$ is also upward closed. Using the above estimate, and recalling that $n \notin R'$, we reach the contradiction

$$0 \leq \sum_{l \in R' \cup \{n\}} (\nu_l - \mu_l) = \sum_{l \in R'}(\nu_l - \mu_l) + (\nu_n - \mu_n) \leq 2\epsilon + (\nu_n - \mu_n) = \frac{1}{3}(\nu_n - \mu_n) < 0. \blacktriangleleft$$

Nawrotzki's algorithm for the proof of Theorem 2 proceed in three steps:

1. Start with the diagonal matrix built from μ.
2. Iteratively modify this matrix in such a way, that the set of all points (n, m) where all of Equation (7)–Equation (9) fail (for certain modified sequences), gets smaller in each step.
3. Pass to the limit, so to reach a situation where Proposition 5 applies.

The single steps of the recursive process **2.** are realised by maps which act on $\ell^1(\mathbb{N}\times\mathbb{N})$. To define those maps, we first introduce an abbreviation for row- and column sums of a matrix. Given $\Lambda = (\lambda_{n,m})_{n,m\in\mathbb{N}} \in \ell^1(\mathbb{N}\times\mathbb{N})$, we denote

$$\lambda_{*,m} := \sum_{n\in\mathbb{N}} \lambda_{n,m}, \quad \lambda_{n,*} := \sum_{m\in\mathbb{N}} \lambda_{n,m}.$$

Note that these series converge absolutely since $\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$.

▶ **Definition 7.** *Let $\nu = (\nu_n)_{n\in\mathbb{N}} \in \ell^1(\mathbb{N})$ and $(n,m) \in \mathbb{N}\times\mathbb{N}$. We define maps*

$$\alpha^\nu_{n,m}\colon \ell^1(\mathbb{N}\times\mathbb{N}) \to [0,\infty), \quad \Phi^\nu_{n,m}\colon \ell^1(\mathbb{N}\times\mathbb{N}) \to \ell^1(\mathbb{N}\times\mathbb{N}).$$

- *For $\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$ set*

$$\alpha^\nu_{n,m}(\Lambda) := \min\Big\{\lambda_{*,n} - \nu_n, \nu_m - \lambda_{*,m}, \inf_{R\in\mathcal{R}_{n,m}} \sum_{l\in R}(\nu_l - \lambda_{*,l})\Big\},$$

  if $n \preccurlyeq m$ and this minimum is positive, and set $\alpha^\nu_{n,m} := 0$ otherwise.
- *For $\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$ let $\Phi^\nu_{n,m}(\Lambda)$ be the matrix with the entries*

$$\Big[\Phi^\nu_{n,m}\Big]_{l,k}(\Lambda) := \begin{cases} \lambda_{l,k} - \alpha^\nu_{n,m}(\Lambda) & \text{if } (l,k) = (n,n), \\ \lambda_{l,k} + \alpha^\nu_{n,m}(\Lambda) & \text{if } (l,k) = (n,m), \\ \lambda_{l,k} & \text{otherwise.} \end{cases}$$

Note that $\Phi^\nu_{n,m}$ is well-defined, since $\alpha^\nu_{n,m} \neq 0$ implies that $n \neq m$, and since it is obvious that $\Phi^\nu_{n,m}(\Lambda)$ is again summable.

Let us collect some more obvious properties of the transformations $\Phi^\nu_{n,m}$.

▶ Remark 8. For each $\nu \in \ell^1(\mathbb{N})$ and $(n,m) \in \mathbb{N}\times\mathbb{N}$, the following statements hold.

1. $\operatorname{supp}\Phi^\nu_{n,m}(\Lambda) \subseteq \big(\operatorname{supp}\Lambda\big) \cup \{(n,n),(n,m)\}$,
2. $\forall l \in \mathbb{N}.\ \Big[\Phi^\nu_{n,m}(\Lambda)\Big]_{l,*} = \lambda_{l,*}$,
3. $\forall l \in \mathbb{N}.\ \Big[\Phi^\nu_{n,m}(\Lambda)\Big]_{*,l} = \begin{cases} \lambda_{*,l} - \alpha^\nu_{n,m}(\Lambda) & \text{if } l = n, \\ \lambda_{*,l} + \alpha^\nu_{n,m}(\Lambda) & \text{if } l = m, \\ \lambda_{*,l} & \text{otherwise.} \end{cases}$

Having $\alpha^\nu_{n,m}(\Lambda) = 0$ just means that at the point (n,m) one of Equation (7)–Equation (9) holds for the sequences $(\lambda_{*,n})_{n\in\mathbb{N}}$ and $(\nu_n)_{n\in\mathbb{N}}$. Moreover, in this case, $\Phi^\nu_{n,m}$ does not change Λ. We are interested to see what happens if $\alpha^\nu_{n,m}(\Lambda) > 0$.

▶ **Definition 9.** *Let $\nu \in \ell^1(\mathbb{N})$ and $\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$. Then we set*

$$S(\Lambda) := \Big\{(n,m) \in \mathbb{N}\times\mathbb{N} \mid\ \alpha^\nu_{n,m}(\Lambda) > 0\Big\}.$$

Moreover, we denote by $\pi_1(S(\Lambda))$ and $\pi_2(S(\Lambda))$ the projections of $S(\Lambda)$ onto the first and second, respectively, component.

To avoid bulky notation, we do not explicitly notate the dependency on ν. Moreover, observe that $S(\Lambda)$ is contained in $\preccurlyeq$ and does not intersect the diagonal, in fact,

$$\pi_1(S(\Lambda)) \cap \pi_2(S(\Lambda)) = \emptyset.$$

In the next proposition we show that $\Phi^\nu_{n,m}$ preserves several relevant properties and indeed shrinks the set $S(\Lambda)$.

▶ **Proposition 10.** *Let* $\nu = (\nu_n)_{n\in\mathbb{N}} \in \ell^1(\mathbb{N})$, $\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$, *and assume that*

$$\forall n, m \in \mathbb{N}.\ \lambda_{n,m} \geq 0 \quad \textit{and} \quad \sum_{n,m\in\mathbb{N}} \lambda_{n,m} = 1, \tag{10}$$

$$\forall n \in \pi_1(S(\Lambda)).\ \lambda_{*,n} = \lambda_{n,n}. \tag{11}$$

$$\forall R \subseteq \mathbb{N} \textit{ upward closed w.r.t. } \preccurlyeq.\ \sum_{l\in R} \lambda_{*,l} \leq \sum_{l\in R} \nu_l, \tag{12}$$

Further, let $(n', m') \in \mathbb{N}\times\mathbb{N}$, *and assume that* $\alpha^{\nu}_{n',m'}(\Lambda) > 0$. *Then*

1. $\Phi^{\nu}_{n',m'}(\Lambda)$ *satisfies Equation* (10), *Equation* (11), *and Equation* (12),
2. $S\big(\Phi^{\nu}_{n',m'}(\Lambda)\big) \subseteq S(\Lambda) \setminus \{(n', m')\}$.

Proof. To shorten notation, we write

$$\Lambda' = (\lambda'_{n,m})_{n,m\in\mathbb{N}} := \Phi^{\nu}_{n',m'}(\Lambda).$$

We start with showing that Λ' satisfies Equation (10) and Equation (12). Let $(n, m) \neq (n', n')$. Then $\lambda'_{n,m} \geq \lambda_{n,m}$ and hence is nonnegative. For $(n, m) = (n', n')$ we use (11) to obtain

$$\lambda'_{n',n'} = \lambda_{n',n'} - \alpha^{\nu}_{n',m'}(\Lambda) = \lambda_{*,n'} - \alpha^{\nu}_{n',m'}(\Lambda) \geq \nu_{n'} \geq 0.$$

Obviously, applying $\Phi^{\nu}_{n',m'}$ does not change the total sums of the entries of a matrix. Thus

$$\sum_{n,m\in\mathbb{N}} \lambda'_{n,m} = \sum_{n,m\in\mathbb{N}} \lambda_{n,m} = 1.$$

We see that Equation (10) holds.

Let $R \subseteq \mathbb{N}$ be upward closed. If $R \notin \mathcal{R}_{n',m'}$, then

$$\sum_{l\in R} \lambda'_{*,l} \leq \sum_{l\in R} \lambda_{*,l} \leq \sum_{l\in R} \nu_l.$$

Next, for $R \in \mathcal{R}_{n',m'}$

$$\sum_{l\in R} \lambda'_{*,l} = \sum_{l\in R} \lambda_{*,l} + \alpha^{\nu}_{n',m'}(\Lambda), \tag{13}$$

and from this we find

$$\sum_{l\in R} \lambda'_{*,l} = \sum_{l\in R} \lambda_{*,l} + \alpha^{\nu}_{n',m'}(\Lambda) \leq \sum_{l\in R} \lambda_{*,l} + \sum_{l\in R} (\nu_n - \lambda_{*,l}) = \sum_{l\in R} \nu_l.$$

Thus Equation (12) holds.

Now we come to the proof of **2.**. This is the major part of the argument.

In the first step we show that $(n', m') \notin S(\Lambda')$. We make a case distinction according to which term is the minimum in the definition of $\alpha^{\nu}_{n',m'}(\Lambda)$.

- Case $\alpha^{\nu}_{n',m'}(\Lambda) = \lambda_{*,n'} - \nu_{n'}$:
  Then $\lambda'_{*,n'} = \nu_{n'}$, and hence $n' \notin \pi_1(S(\Lambda'))$. In particular, $(n', m') \notin S(\Lambda')$.
- Case $\alpha^{\nu}_{n',m'}(\Lambda) = \nu_{m'} - \lambda_{*,n'}$:
  Then $\lambda'_{*,m'} = \nu_{m'}$, and hence $m' \notin \pi_2(S(\Lambda'))$. In particular, $(n', m') \notin S(\Lambda')$.

- Case $\alpha^{\nu}_{n',m'}(\Lambda) = \inf_{\mathcal{R}_{n',m'}} \sum_{l\in R}(\nu_l - \lambda_{*,l})$:
  Recalling Equation (13), we find

$$\inf_{R\in\mathcal{R}_{n',m'}} \sum_{l\in R}(\nu_l - \lambda'_{*,l}) = \inf_{R\in\mathcal{R}_{n',m'}} \sum_{l\in R}\Big[(\nu_l - \lambda_{*,l}) - \alpha^{\nu}_{n',m'}(\Lambda)\Big] = 0.$$

  Thus also in this case $(n',m') \notin S(\Lambda')$.

In the second step, we show that $S(\Lambda') \subseteq S(\Lambda)$. Assume towards a contradiction that $(n,m) \in S(\Lambda') \setminus S(\Lambda)$. Explicitly this means that

$$n \prec m \;\wedge\; \lambda'_{*,n} > \nu_n \;\wedge\; \lambda'_{*,m} < \nu_m \;\wedge\; \inf_{R\in\mathcal{R}_{n,m}} \sum_{l\in R}(\nu_l - \lambda'_{*,l}) > 0$$
$$\wedge\; \Big[\; \lambda_{*,n} \le \nu_n \;\vee\; \lambda_{*,m} \ge \nu_m \;\vee\; \inf_{R\in\mathcal{R}_{n,m}} \sum_{l\in R}(\nu_l - \lambda_{*,l}) = 0 \;\Big]$$

We distinguish cases according to the disjunction in the square bracket.

- Case $\lambda_{*,n} \le \nu_n$:
  The sum of the n-th column increases, and thus we must have $n = m'$. This implies

$$\lambda'_{*,n} = \lambda'_{*,m'} = \lambda_{*,m'} + \alpha^{\nu}_{n',m'}(\Lambda) \le \nu_{m'} = \nu_n,$$

  which contradicts the second term in the conjunction.
- Case $\lambda_{*,m} \ge \nu_m$:
  The sum of the m-th column decreases, and thus we must have $m = n'$. This implies

$$\lambda'_{*,m} = \lambda'_{*,n'} = \lambda_{*,n'} - \alpha^{\nu}_{n',m'}(\Lambda) \ge \nu_{n'} = \nu_m,$$

  which contradicts the third term in the conjunction.
- Case $\inf_{R\in\mathcal{R}_{n,m}} \sum_{l\in R}(\nu_l - \lambda_{*,l}) = 0$:
  Choose $R' \in \mathcal{R}_{n,m}$ such that

$$\sum_{l\in R'}(\nu_l - \lambda_{*,l}) < \inf_{R\in\mathcal{R}_{n,m}} \sum_{l\in R}(\nu_l - \lambda'_{*,l}).$$

  Then, in particular, the value of the sum over all $l \in R'$ decreases, and we must have $n' \in R'$ and $m' \notin R'$. Since R' is upward closed and $n' \prec m'$, this is a contradiction.

The proof of **2.** is complete.

It remains to deduce Equation (11). Let $n \in \pi_1(S(\Lambda'))$. Then also $n \in \pi_1(S(\Lambda))$, and therefore $n \ne m'$ and $\lambda_{*,n} = \lambda_{n,n}$. From the first property we obtain that the n-th column is modified at most at its diagonal entry, and now the second implies that $\lambda'_{*,n} = \lambda'_{n,n}$. ◀

Next, we investigate iterative application of maps $\Phi^{\nu}_{n,m}$. Start with $\nu \in \ell^1(\mathbb{N})$, $\Lambda^{(0)} \in \ell^1(\mathbb{N}\times\mathbb{N})$, and a sequence $((n_k,m_k))_{k\ge1}$ of points in $\mathbb{N}\times\mathbb{N}$. From this data, we built the sequence $(\Lambda^{(k)})_{k\in\mathbb{N}}$ where

$$\Lambda^{(k)} := \Big[\Phi^{\nu}_{n_k,m_k} \circ \cdots \circ \Phi^{\nu}_{n_1,m_1}\Big]\big(\Lambda^{(0)}\big). \tag{14}$$

It turns out that, in the situation of Theorem 2, sequences of this form converge. In fact, they do so because of a very simple reason, namely, monotonicity.

▶ **Lemma 11.** *Let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ be a sequence in $\ell^1(\mathbb{N}\times\mathbb{N})$, such that*

$$\sup_{k\in\mathbb{N}} \|\Lambda^{(k)}\|_1 < \infty, \quad \forall n,m,k\in\mathbb{N}.\ \lambda^{(k)}_{n,m} \geq 0,$$

and that there exists a partition $\mathbb{N}\times\mathbb{N} = A\dot\cup B$ such that $(\lambda^{(k)}_{n,m})_{k\in\mathbb{N}}$ is nondecreasing for all $(n,m)\in A$ and nonincreasing for all $(n,m)\in B$.

Then the limit $\Lambda := \lim_{k\to\infty}\Lambda^{(k)}$ exists in the ℓ^1-norm.

Proof. Each of the sequences $(\lambda^{(k)}_{n,m})_{k\in\mathbb{N}}$ is monotone and bounded, hence convergent. Denote $\lambda_{n,m} := \lim_{k\to\infty}\lambda^{(k)}_{n,m}$. We have to show that the pointwise limit $\Lambda = (\lambda_{n,m})_{n,m\in\mathbb{N}}$ is actually attained in the ℓ^1-norm. To this end we split the corresponding sum according to the given partition.

For each $(n,m)\in A$ the sequence $(\lambda^{(k)}_{n,m})_{k\in\mathbb{N}}$ is nondecreasing, and hence the monotone convergence theorem yields

$$\sum_{(n,m)\in A}\lambda_{n,m} = \lim_{k\to\infty}\sum_{(n,m)\in A}\lambda^{(k)}_{n,m} \leq \sup_{k\in\mathbb{N}}\|\Lambda^{(k)}\|_1 < \infty.$$

Since $\lambda_{n,m}\geq\lambda_{n,m}-\lambda^{(k)}_{n,m}\geq 0$, we may now refer to the bounded convergence theorem to obtain that

$$\lim_{k\to\infty}\sum_{(n,m)\in A}\left|\lambda^{(k)}_{n,m}-\lambda_{n,m}\right| = 0.$$

For each $(n,m)\in B$ and $k\in\mathbb{N}$ we have

$$\lambda^{(0)}_{n,m}\geq\lambda^{(k)}_{n,m}\geq\lambda^{(k)}_{n,m}-\lambda_{n,m}\geq 0.$$

Since $\sum_{(n,m)\in B}\lambda^{(0)}_{n,m}<\infty$, the bounded convergence theorem applies, and we find that

$$\lim_{k\to\infty}\sum_{(n,m)\in B}\left|\lambda^{(k)}_{n,m}-\lambda_{n,m}\right| = 0. \qquad ◀$$

▶ **Corollary 12.** *Assume that $\Lambda^{(0)}$ satisfies Equation* (10) *and Equation* (11)*, let $((n_k,m_k))_{k\geq 1}$ be any sequence, and let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ be defined by Equation* (14)*. Then the limit*

$$\Lambda := \lim_{k\to\infty}\Lambda^{(k)}$$

exists w.r.t. the ℓ^1-norm.

Proof. Since $\alpha^{\nu}_{n,m}(\Lambda)$ is always nonnegative, a partition of $\mathbb{N}\times\mathbb{N}$ required to apply Lemma 11 is obtained by taking the diagonal as the set A. ◀

Now we show that, when passing to a limit, the set $S(\Lambda)$ can be controlled.

▶ **Lemma 13.** *Let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ be a sequence in $\ell^1(\mathbb{N}\times\mathbb{N})$ which converges in the ℓ^1-norm, and denote $\Lambda := \lim_{k\to\infty}\Lambda^{(k)}$. Then*

$$S(\Lambda)\subseteq\bigcup_{N\in\mathbb{N}}\bigcap_{k\geq N}S(\Lambda^{(k)}).$$

Proof. Let $(n,m) \in S(\Lambda)$, and set $\epsilon := \frac{1}{2}\alpha^{\nu}_{n,m}(\Lambda)$. Choose $N \in \mathbb{N}$ such that

$$\forall k \geq N.\ \|\Lambda^{(k)} - \Lambda\|_1 \leq \epsilon.$$

Then for all $k \geq N$

$$\lambda^{(k)}_{*,n} \geq \lambda_{*,n} - \epsilon \geq \nu_n, \quad \lambda^{(k)}_{*,m} \leq \lambda_{*,m} + \epsilon \leq \nu_m,$$

and for all $R \in \mathcal{R}_{n,m}$

$$\sum_{l\in R} \big(\nu_l - \lambda^{(k)}_{*,l}\big) \geq \sum_{l\in R} (\nu_l - \lambda_{*,l}) - \epsilon \geq \epsilon > 0$$

Thus $(n,m) \in S(\Lambda^{(k)})$. ◀

We have collected all the neccessary tools needed for the proof of Theorem 2.

Proof of Theorem 2. Let μ, ν, and $\preccurlyeq$, be given, and assume that Equation (1) and Equation (2) hold.

Let $\Lambda^{(0)} = (\lambda^{(0)}_{n,m})_{n,m\in\mathbb{N}}$ be the diagonal matrix built from μ, i.e.,

$$\lambda^{(0)}_{n,m} := \begin{cases} \mu_n & \text{if } n = m, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Choose a sequence of points $((n_k, m_k))_{k\geq 1}$ in $\mathbb{N}\times\mathbb{N}$ which covers $\prec$. For example, every enumeration of $\mathbb{N}\times\mathbb{N}$ certainly has this property. Now define $\Lambda^{(k)}$ by Equation (14) using this sequence.

By Proposition 10, each $\Lambda^{(k)}$ satisfies Equation (10), Equation (11), and Equation (12). Moreover,

$$S(\Lambda^{(k)}) \subseteq S(\Lambda^{(0)}) \setminus \big\{(n_1,m_1),\dots,(n_k,m_k)\big\}.$$

The limit

$$\Lambda = (\lambda_{n,m})_{n,m\in\mathbb{N}} := \lim_{k\to\infty} \Lambda^{(k)}$$

exists in the ℓ^1-norm by Corollary 12, and $S(\Lambda) = \emptyset$ by Lemma 13.

Clearly, Equation (3)–Equation (5) hold for Λ. By virtue of Proposition 10, we may apply Proposition 5 with the sequences $(\lambda_{*,n})_{n\in\mathbb{N}}$ and $(\nu_n)_{n\in\mathbb{N}}$, and obtain that also Equation (6) holds. ◀

We refer to the procedure carried out in this proof as *Nawrotzki's algorithm* being *performed along* the sequence $((n_k,m_k))_{k\geq 1}$.

▶ Remark 14. For later use, we observe the following fact. Let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ be a sequence produced by an application of Nawrotzki's algorithm. Then off-diagonal elements $\lambda^{(k)}_{n,m}$ change their value at most once when k runs through $\mathbb{N}$. Namely, only when $(n,m) = (n_k,m_k)$ and it happens that $\alpha^{\nu}_{n,m}(\Lambda^{(k-1)}) > 0$.

3 A constructive variant of the algorithm

Nawrotzki's proof of Theorem 2 is non-constructive for the following reason:

- The set $\mathcal{R}_{n,m}$ is in general infinite, and its elements themselves are in general infinite.

Because of this, computing the numbers $\alpha^{\nu}_{n,m}$ requires to evaluate the sum of infinite series and an infimum of an infinite set. Hence, it is not possible to compute any term of the sequence $(\Lambda^{(k)})_{k\in\mathbb{N}}$, which converges to a solution matrix Λ, with a finite number of algebraic operations.

Our aim is to give a proof of Theorem 2 which is more constructive in the following sense.

▶ **Theorem 15.** *Let $\mu, \nu, \preccurlyeq$ be given such that Equation (1) and Equation (2) hold. Then there exists a sequence $(\Delta^{(k)})_{k\in\mathbb{N}}$ of matrices in $\ell^1(\mathbb{N}\times\mathbb{N})$ with the following properties.*

1. *Each $\Delta^{(k)}$ can be computed from the given data μ and ν by a finite number of algebraic operations.*
2. *The limit $\Delta := \lim_{k\to\infty}\Delta^{(k)}$ exists in the ℓ^1-norm and satisfies Equation (3)–Equation (6).*

As usual we use the notation $\Delta^{(k)} = (\delta^{(k)}_{n,m})_{n,m\in\mathbb{N}}$ and $\Delta = (\delta_{n,m})_{n,m\in\mathbb{N}}$.

3. *For each fixed $(n,m)\in\mathbb{N}\times\mathbb{N}$ with $n \prec m$, and for each $\epsilon > 0$, a number k_0 with the property that*

$$\forall k \geq k_0.\ |\delta^{(k)}_{n,m} - \delta_{n,m}| \leq \epsilon$$

can be computed from the given data μ and ν by a finite number of algebraic operations

While the speed of pointwise convergence is controlled by the assertion in item **3.** (even in a constructive way), we have no control of the speed of ℓ^1-convergence.

The idea to prove this theorem is the simplest possible: we consider cut-off data μ_N, ν_N instead of μ, ν, apply Nawrotzki's algorithm to the truncated data, and then send the cut-off point to infinity. Realising this idea, however, requires some work.

We start with discussing convergence matters. The error when using cut-off's instead of the full data can be controlled using the following general perturbation lemma.

▶ **Lemma 16.** *Let $\nu, \tilde\nu \in \ell^1(\mathbb{N})$, $\Lambda, \tilde\Lambda \in \ell^1(\mathbb{N}\times\mathbb{N})$, and $(n,m)\in\mathbb{N}\times\mathbb{N}$. Then*

$$\left|\alpha^{\nu}_{n,m}(\Lambda) - \alpha^{\tilde\nu}_{n,m}(\tilde\Lambda)\right| \leq \|\Lambda - \tilde\Lambda\|_1 + \|\nu - \tilde\nu\|_1. \tag{16}$$

Proof. We have

$$\left|(\lambda_{*,n} - \nu_n) - (\tilde\lambda_{*,n} - \tilde\nu_n)\right| \leq \sum_{l\in\mathbb{N}} |\lambda_{l,n} - \tilde\lambda_{l,n}| + |\nu_n - \tilde\nu_n| \leq \|\Lambda - \tilde\Lambda\|_1 + \|\nu - \tilde\nu\|_1,$$

and in the same way

$$\left|(\lambda_{*,m} - \nu_m) - (\tilde\lambda_{*,m} - \tilde\nu_m)\right| \leq \sum_{l\in\mathbb{N}} |\lambda_{l,m} - \tilde\lambda_{l,m}| + |\nu_m - \tilde\nu_m| \leq \|\Lambda - \tilde\Lambda\|_1 + \|\nu - \tilde\nu\|_1.$$

Next let $R\subseteq\mathbb{N}$. Then

$$\left|\sum_{l\in R}(\nu_l - \lambda_{*,l}) - \sum_{l\in R}(\tilde\nu_l - \tilde\lambda_{*,l})\right| \leq \leq \sum_{l\in R}\sum_{k\in\mathbb{N}} |\lambda_{k,l} - \tilde\lambda_{k,l}| + \sum_{l\in R}|\nu_l - \tilde\nu_l| \leq \|\Lambda - \tilde\Lambda\|_1 + \|\nu - \tilde\nu\|_1.$$

It follows that

$$\begin{aligned}\Big|\inf\Big(\{\lambda_{*,n}-\nu_n,\nu_m-\lambda_{*,m}\}\cup\Big\{\sum_{l\in R}(\nu_l-\lambda_{*,l})\mid R\in\mathcal{R}_{n,m}\Big\}\Big)\\ -\inf\Big(\{\tilde\lambda_{*,n}-\tilde\nu_n,\tilde\nu_m-\tilde\lambda_{*,m}\}\cup\Big\{\sum_{l\in R}(\tilde\nu_l-\tilde\lambda_{*,l})\mid R\in\mathcal{R}_{n,m}\Big\}\Big)\Big|\\ \le\|\Lambda-\tilde\Lambda\|_1+\|\nu-\tilde\nu\|_1.\end{aligned}$$

This is Equation (16) if $n\preccurlyeq m$. Otherwise $\alpha^{\nu}_{n,m}=\alpha^{\tilde\nu}_{n,m}(\tilde\Lambda)=0$, and the required estimate holds trivially. ◀

▶ **Corollary 17.** *Let $\nu,\tilde\nu\in\ell^1(\mathbb{N})$, $\Lambda,\tilde\Lambda\in\ell^1(\mathbb{N}\times\mathbb{N})$, and $((n_k,m_k))_{k\ge1}$ be a sequence in $\mathbb{N}\times\mathbb{N}$. Let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ and $(\tilde\Lambda^{(k)})_{k\in\mathbb{N}}$ be the sequences defined by Equation (14) starting from $\Lambda^{(0)}:=\Lambda$ and $\tilde\Lambda^{(0)}:=\tilde\Lambda$, respectively. Moreover, set*

$$\epsilon:=\|\Lambda-\tilde\Lambda\|_1+\|\nu-\tilde\nu\|_1.$$

Then

$$\forall k\in\mathbb{N}.\ \|\Lambda^{(k)}-\tilde\Lambda^{(k)}\|_1+\|\nu-\tilde\nu\|_1\le 3^k\epsilon.$$

Proof. For $k=0$ this is the definition of ϵ. Then proceed inductively based on the estimate

$$\big\|\Phi^{\nu}_{n,m}(\Lambda)-\Phi^{\tilde\nu}_{n,m}(\tilde\Lambda)\big\|_1\le\|\Lambda-\tilde\Lambda\|_1+2|\alpha^{\nu}_{n,m}(\Lambda)-\alpha^{\tilde\nu}_{n,m}(\tilde\Lambda)|,$$

which holds for all $\nu,\tilde\nu,\Lambda,\tilde\Lambda,n,m$. ◀

Now we turn to computability matters. To settle these, we need one more notation.

▶ **Definition 18.** *Let $L\subseteq\mathbb{N}$, and $n,m\in L$ with $n\prec m$. Then we set*

$$\mathcal{R}^L_{n,m}:=\big\{R\subseteq L\mid n\notin R,m\in R,\forall k\in R,l\in L.\ k\preccurlyeq l\Rightarrow l\in R\big\}.$$

▶ **Lemma 19.** *Let $\nu\in\ell^1(\mathbb{N})$, $\Lambda\in\ell^1(\mathbb{N}\times\mathbb{N})$, let $L\subseteq\mathbb{N}$, and assume that*

$$\operatorname{supp}\nu\subseteq L,\quad \operatorname{supp}\Lambda\subseteq L\times L. \tag{17}$$

Then

$$\forall(n,m)\notin L\times L.\ \alpha^{\nu}_{n,m}(\Lambda)=0, \tag{18}$$

$$\forall(n,m)\in\mathbb{N}\times\mathbb{N}.\ \operatorname{supp}\Phi^{\nu}_{n,m}(\Lambda)\subseteq L\times L, \tag{19}$$

$$\forall n,m\in L,n\prec m.\ \inf_{R\in\mathcal{R}_{n,m}}\sum_{l\in R}(\nu_l-\lambda_{*,l})=\inf_{R\in\mathcal{R}^L_{n,m}}\sum_{l\in R}(\nu_l-\lambda_{*,l}). \tag{20}$$

Proof. The assumption on the supports of ν and Λ shows that

$$\forall n\notin L.\ \nu_n=\lambda_{*,n}=0.$$

From this Equation (18), and in turn also Equation (19), follows. Moreover, for every subset $R\subseteq\mathbb{N}$

$$\sum_{l\in R}(\nu_l-\lambda_{*,l})=\sum_{l\in R\cap L}(\nu_l-\lambda_{*,l}).$$

To establish Equation (20), we show that for all $n, m \in L$ with $n \prec m$

$$\mathcal{R}^L_{n,m} = \{R \cap L \mid\ R \in \mathcal{R}_{n,m}\}.$$

The inclusion "$\supseteq$" is clear. For the reverse inclusion observe that, for each $R \in \mathcal{R}^L_{n,m}$, the set

$$R' := \big\{l \in \mathbb{N} \mid\ \exists k \in R.\ k \preccurlyeq l\big\}$$

belongs to $\mathcal{R}_{n,m}$ and $R' \cap L = R$. ◀

▶ **Corollary 20.** *Let $\nu \in \ell^1(\mathbb{N})$ and $\Lambda \in \ell^1(\mathbb{N} \times \mathbb{N})$ be finitely supported. Then*

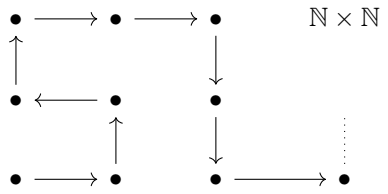
1. *for each $n \in \mathbb{N}$ the number $\lambda_{*,n}$ is a finite sum, and*
2. *for each $(n, m) \in \mathbb{N} \times \mathbb{N}$ the infimum in the definition of $\alpha^{\nu}_{n,m}(\Lambda)$ is the minimum of a finite number of finite sums.*

Proof. We can choose a finite set $L \subseteq \mathbb{N}$ such that Equation (17) holds. Then each set $\mathcal{R}^L_{n,m}$, and also each of its elements, is finite. ◀

Proof of Theorem 15. Consider truncated data: for $N \in \mathbb{N}$, let $\mu_N = (\mu_{N;n})_{n\in\mathbb{N}}$ and $\nu_N = (\nu_{N;n})_{n\in\mathbb{N}}$ be defined by

$$\mu_{N;n} := \begin{cases} \mu_n & \text{if } n < N, \\ 1 - \sum_{l<N} \mu_l & \text{if } n = N, \\ 0 & \text{if } n > N, \end{cases} \qquad \nu_{N;n} := \begin{cases} \nu_n & \text{if } n < N, \\ 1 - \sum_{l<N} \nu_l & \text{if } n = N, \\ 0 & \text{if } n > N. \end{cases}$$

We execute Nawrotzki's algorithm with the data μ_N, ν_N along the enumeration $((n_k, m_k))_{k\geq 1}$ of $\mathbb{N} \times \mathbb{N}$ which is defined by running through the scheme



and dropping all points (n, m) which do not satisfy $n \prec m$.

This provides us with sequences $(\Lambda^{(k)}_N)_{k\in\mathbb{N}}$, $N \in \mathbb{N}$. According to Lemma 19 and Corollary 20, we have

$$\operatorname{supp} \Lambda^{(k)}_N \subseteq \{0, \dots, N\} \times \{0, \dots, N\},$$

and each $\Lambda^{(k)}_N$ can be computed by a finite number of algebraic operations.

Let $(\Lambda^{(k)})_{k\in\mathbb{N}}$ be the sequence obtained by running Nawrotzki's algorithm along the same sequence $((n_k, m_k))_{k\geq 1}$ but starting with the full data μ, ν. We have

$$\|\Lambda^{(0)} - \Lambda^{(0)}_N\|_1 = 2 \sum_{n>N} \mu_n, \quad \|\nu - \nu\|_1 = 2 \sum_{n>N} \nu_n,$$

and hence

$$\|\Lambda^{(0)} - \Lambda^{(0)}_N\|_1 + \|\nu - \nu\|_1 = 2 \sum_{n>N} (\mu_n + \nu_n) = 2\Big(2 - \sum_{n\leq N} (\mu_n + \nu_n)\Big) =: \epsilon_N.$$

Corollary 17 applies and leads to the basic estimate

$$\forall k \in \mathbb{N}, N \in \mathbb{N}.\ \|\Lambda^{(k)} - \Lambda^{(k)}_N\|_1 + \|\nu - \nu\|_1 \leq 3^k \epsilon_N. \tag{21}$$

The next step is to define a sequence $(\Delta_k)_{k\in\mathbb{N}}$. This is done as follows: given $k \in N$, choose $N_k \in \mathbb{N}$ with

$$\epsilon_{N_k} \leq \frac{1}{k \cdot 3^k},$$

and set $\Delta_k := \Lambda^{(k)}_{N_k}$.

The number N_k can be found in finitely many steps by summing up beginning sections of μ and ν. Together with what we already observed above, thus, each Δ_k can be computed in finitely many steps.

We know that the limit $\Lambda := \lim_{k\to\infty} \Lambda^{(k)}$ exists in the ℓ^1-norm and satisfies Equation (3) – Equation (6). The basic estimate Equation (21) yields

$$\|\Lambda^{(k)} - \Delta^{(k)}\|_1 \leq \frac{1}{k},$$

and we see that also $\lim_{k\to\infty} \Delta^{(k)} = \Lambda$ in the ℓ^1-norm.

Let $(n, m) \in \mathbb{N} \times \mathbb{N}$ with $n \prec m$ and $\epsilon > 0$ be given. Define $k_0 \in \mathbb{N}$ as the least integer larger or equal to

$$\max\left\{\frac{1}{\epsilon}, \big(\max\{n, m\}\big)^2\right\}.$$

Then $(n, m) \in \{(n_1, m_1), \ldots, (n_{k_0}, m_{k_0})\}$ and for all $k \geq k_0$

$$\|\Lambda^{(k)} - \Delta^{(k)}\|_1 \leq \epsilon.$$

Now recall Remark 14: the entry $\lambda^{(k)}_{n,m}$ is constant for $k \geq k_0$. This implies that, for all $k \geq k_0$,

$$|\lambda_{n,m} - \delta^{(k)}_{n,m}| = |\lambda^{(k)}_{n,m} - \delta^{(k)}_{n,m}| \leq \|\Lambda^{(k)} - \Delta^{(k)}\|_1 \leq \epsilon.$$

The proof of Theorem 15 is complete. ◀

References

1. G. Barthe, T. Espitau, J. Hsu, T. Sato, and P.-Y. Strub. *-liftings for differential privacy. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 102:1–102:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.102.
2. G. Barthe, T. Espitau, J. Hsu, T. Sato, and P.-Y. Strub. Relational ⋆⋆\star-liftings for differential privacy. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:18)2019.
3. G. Barthe, M. Gaboardi, B. Grégoire, J. Hsu, and P.-Y. Strub. Proving differential privacy via probabilistic couplings. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 749–758. ACM, 2016. doi:10.1145/2933575.2934554.
4. M. Beiglböck and N. Juillet. On a problem of optimal transport under marginal martingale constraints. *Ann. Probab.*, 44(1):42–106, 2016. doi:10.1214/14-AOP966.
5. P. Berti, L. Pratelli, P. Rigo, and F. Spizzichino. Equivalent or absolutely continuous probability measures with given marginals. *Depend. Model.*, 3(1):47–58, 2015. doi:10.1515/demo-2015-0004.

6 E. D'Aniello and J.D.M. Wright. Finding measures with given marginals. *Q. J. Math.*, 51(4):405–416, 2000. doi:10.1093/qjmath/51.4.405.

7 M.H.A. Davis and D.G. Hobson. The range of traded option prices. *Math. Finance*, 17(1):1–14, 2007. doi:10.1111/j.1467-9965.2007.00291.x.

8 S. Friedland, J. Ge, and L. Zhi. Quantum Strassen's theorem. *Infin. Dimens. Anal. Quantum Probab. Relat. Top.*, 23(3):2050020, 29, 2020. doi:10.1142/S0219025720500204.

9 N. Gaffke and L. Rüschendorf. On the existence of probability measures with given marginals. *Statist. Decisions*, 2(1-2):163–174, 1984.

10 J. Hsu. *Probabilistic couplings for probabilistic reasoning*. Phd thesis, University of Pennsylvania, 2017. URL: https://repository.upenn.edu/edissertations/3017/.

11 C. Jones. *Probabilistic Non-determinism*. Phd thesis, University of Edinburgh, 1989.

12 A. Jung and R. Tix. The troublesome probabilistic powerdomain. In *Third Workshop on Computation and Approximation (Comprox III) (Birmingham, 1997)*, volume 13 of *Electron. Notes Theor. Comput. Sci.*, page 22. Elsevier Sci. B. V., Amsterdam, 1998.

13 T. Kamae, U. Krengel, and G.L. O'Brien. Stochastic inequalities on partially ordered spaces. *Ann. Probability*, 5(6):899–912, 1977. doi:10.1214/aop/1176995659.

14 J. Kawabe. A type of Strassen's theorem for positive vector measures with values in dual spaces. *Proc. Amer. Math. Soc.*, 128(11):3291–3300, 2000. doi:10.1090/S0002-9939-00-05384-3.

15 H.G. Kellerer. Funktionen auf Produkträumen mit vorgegebenen Marginal-Funktionen. *Math. Ann.*, 144:323–344, 1961. doi:10.1007/BF01470505.

16 H.G. Kellerer. Duality theorems for marginal problems. *Z. Wahrsch. Verw. Gebiete*, 67(4):399–432, 1984. doi:10.1007/BF00532047.

17 H. König. On the marginals of probability contents on lattices. *Mathematika*, 58(2):319–323, 2012. doi:10.1112/S0025579311002427.

18 V.T. Koperberg. Couplings and matchings. Bachelor thesis, Universiteit Leiden, 2016. URL: https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/koperbergbach.pdf.

19 K. Nawrotzki. Eine Monotonieeigenschaft zufälliger Punktfolgen. *Math. Nachr.*, 24:193–200, 1962. doi:10.1002/mana.19620240305.

20 H.J. Skala. The existence of probability measures with given marginals. *Ann. Probab.*, 21(1):136–142, 1993. URL: http://links.jstor.org/sici?sici=0091-1798(199301)21:1<136:TEOPMW>2.0.CO;2-I&origin=MSN.

21 V. Strassen. The existence of probability measures with given marginals. *Ann. Math. Statist.*, 36:423–439, 1965. doi:10.1214/aoms/1177700153.

Minimality Notions via Factorization Systems

Thorsten Wißmann
Radboud University Nijmegen, The Netherlands

Abstract

For the minimization of state-based systems (i.e. the reduction of the number of states while retaining the system's semantics), there are two obvious aspects: removing unnecessary states of the system and merging redundant states in the system. In the present article, we relate the two aspects on coalgebras by defining an abstract notion of minimality.

The abstract notion minimality and minimization live in a general category with a factorization system. We will find criteria on the category that ensure uniqueness, existence, and functoriality of the minimization aspects. The proofs of these results instantiate to those for reachability and observability minimization in the standard coalgebra literature. Finally, we will see how the two aspects of minimization interact and under which criteria they can be sequenced in any order, like in automata minimization.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Coalgebra, Reachability, Observability, Minimization, Factorization System

Digital Object Identifier 10.4230/LIPIcs.CALCO.2021.24

Category (Co)algebraic pearls

Related Version *Full Version*: `https://arxiv.org/abs/2106.07233`

Funding Supported by the NWO TOP project 612.001.852.

Acknowledgements The author thanks Stefan Milius and Jurriaan Rot for inspiring discussions and thanks the referees for their helpful comments.

1 Introduction

Minimization is a standard task in computer science that comes in different aspects and lead to various algorithmic challenges. The task is to reduce the size of a given system while retaining its semantics, and in general there are two aspects of making the system smaller: 1. merge redundant parts of the system that exhibit the same behaviour (*observability*) and 2. omit unnecessary parts (*reachability*). Hopcroft's automata minimization algorithm [21] is an early example: in a given deterministic automaton, 1. states accepting the same language are identified and 2. unreachable states are removed. Moreover, Hopcroft's algorithm runs in quasilinear time; for an automaton with n states, reachability is computed in $\mathcal{O}(n)$ and observability in $\mathcal{O}(n \log n)$.

Since the reachability is a simple depth-first search, it is straightforward to apply it to other system types. On the other hand, it took decades until quasilinear minimization algorithms for observability were developed for other system types such as transition systems [29], labelled transition systems [14, 34], or Markov chains [12, 35]. Though their differences in complexity, the aspects of observability and reachability have very much in common when modelling state-based systems as coalgebras. Then, observability is the task to find the greatest coalgebra quotient and reachability is the task of finding the smallest subcoalgebra containing the initial state, or generally, a distinguished point of interest.

In the present article, we define an abstract notion of minimality and minimization in a category with an $(\mathcal{E}, \mathcal{M})$-factorization system. Such a factorization systems gives rise to a generalized notion of quotients and subobjects, and the minimization is the task of finding

the least quotient resp. subobject. To make this general setting applicable to coalgebras, we show that the category of coalgebras inherits the factorization system from the base category under a mild assumption (namely that the functor preserves $\mathcal{M}$). Dually, a factorization system also lifts to algebras, and even to the Eilenberg-Moore category.

Then, we will present different characterizations of minimality (Figure 4) and then study properties of minimizations, e.g. under which criteria they exist and are unique, rediscovering the respective proofs for reachability and observability for coalgebras in the literature [4, 22]. When combining the two minimization aspects, we discuss under which criteria reachability and observability can be computed in arbitrary order.

The goal of the present work is not only to show the connections between existing minimality notions, but also to provide a series of basic results that can be used when developing new minimization techniques or even new notions of minimality.

Related Work

There is a series of works [10, 9, 11, 30] that studies the minimization of coalgebras by their duality to algebras. In those works, the correspondence between observability in coalgebras and reachability in algebras is used. For instance, Rot [30] relates the final sequence (for observability in coalgebras) with the initial sequence (for reachability in algebras). In the present paper however, we consider both observability and reachability on an abstract level that work for a general factorization system and discuss their instance in coalgebras. The paper is based on Chapter 7 of the author's PhD dissertation [38].

If not included in the main text, detailed proofs of all results can be found in the appendix, both for the standard results recalled in the preliminaries and the new results of the main sections.

2 Preliminaries

In the following, we assume basic knowledge of category theory (cf. standard textbooks [2, 5]).

Given a diagram $D\colon \mathcal{D} \to \mathcal{C}$ (i.e. a functor D from a small category $\mathcal{D}$), we denote its limit by $\lim D$ and colimit by $\operatorname{colim} D$ – if they exist. The limit projections, resp. colimit injections, are denoted by

$$\mathsf{pr}_i\colon \lim D \to Di \qquad \mathsf{inj}_i\colon Di \to \operatorname{colim} D \qquad \text{for } i \in \mathcal{D}.$$

2.1 Coalgebra

We model state-based systems as coalgebras for an endofunctor $F\colon \mathcal{C} \to \mathcal{C}$ on a category $\mathcal{C}$:

▶ **Definition 2.1.** *An* F-coalgebra *(for an endofunctor* $F\colon \mathcal{C} \to \mathcal{C}$*) is a pair* (C, c) *consisting of an object* C *(of* $\mathcal{C}$*) and a morphism* $c\colon C \to FC$ *(in* $\mathcal{C}$*). An* F*-coalgebra morphism* $h\colon (C, c) \to (D, d)$ *between* F*-coalgebras* (C, c) *and* (D, d) *is a morphism* $h\colon C \to D$ *with* $d \cdot h = Fh \cdot c$ *(see Figure 1a on p. 4 for the corresponding commuting diagram).*

Intuitively, the *carrier* C of a coalgebra (C, c) is the state space and the morphism $c\colon C \to FC$ sends states to their possible next states. The functor of choice F defines how these possible next states FC are structured.

▶ **Example 2.2.** Many well-known system-types can be phrased as coalgebras:

- Deterministic automata (without an explicit initial state) are coalgebras for the Set-functor $FX = 2 \times X^A$, where A is the set of input symbols. In an F-coalgebra (C, c), the first component of $c(x)$ denotes the finality of the state $x \in C$ and the second component is the transition function $A \to C$ of the automaton.
- Labelled transition systems are coalgebras for the Set-functor $FX = \mathcal{P}(A \times X)$ and the coalgebra morphisms preserve bisimilarity.
- Weighted systems with weights in a commutative monoid $(M, +, 0)$ (and finite branching) are coalgebras for the *monoid-valued functor* [18, Def. 5.1]

  $$M^{(X)} = \{\mu \colon X \to M \mid \mu(x) = 0 \text{ for all but finitely many } x \in X\}$$

  which sends a map $f \colon X \to Y$ to the map

  $$M^{(f)} \colon M^{(X)} \to M^{(Y)} \qquad M^{(f)}(\mu)(y) = \sum \{\mu(x) \mid x \in X, f(x) = y\}$$

  In an $M^{(-)}$-coalgebra (C, c), the transition weight from state $x \in C$ to $y \in C$ is given by $c(x)(y) \in M$. E.g. one obtains real-valued weighted systems as coalgebras for the functor $(\mathbb{R}, +, 0)^{(-)}$.
- The *bag* functor is defined by $\mathcal{B}X = (\mathbb{N}, +, 0)^{(X)}$. Equivalently, $\mathcal{B}X$ is the set of finite multisets on X. Its coalgebras can be viewed as weighted systems or as transition systems in which there can be more than one transition between two states.
- A wide range of probabilistic and weighted systems can be obtained as coalgebras for respective distribution functors, see e.g. Bartels et al. [8].

▶ **Definition 2.3.** *The category of F-coalgebras and their morphisms is denoted by* $\mathsf{Coalg}(F)$.

Intuitively, the coalgebra morphisms preserve the behaviour of states:

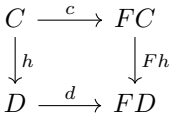
▶ **Definition 2.4.** *In* Set*, two states $x, y \in C$ in an F-coalgebra (C, c) are* behaviourally equivalent *if there is a coalgebra homomorphism $h \colon (C, c) \to (D, d)$ with $h(x) = h(y)$.*

▶ **Example 2.5.**

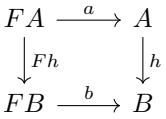
- For deterministic automata ($FX = 2 \times X^A$), states are behaviourally equivalent iff they accept the same language [31, Example 9.5]. Indeed, for sufficiency, if two states x, y in an F-coalgebra are identified by a coalgebra homomorphism, then one can show by induction over input words $w \in A^*$ that either both states or neither of them accepts w. For necessity, note that the map sending states to their semantics $C \to \mathcal{P}(A^*)$ is a coalgebra homomorphism.
- For labelled transition systems ($FX = \mathcal{P}(A \times X)$), states are behaviourally equivalent iff they are bisimilar [1].
- For weighted systems, i.e. coalgebras for $M^{(-)}$, the coalgebraic behavioural equivalence captures weighted bisimilarity [23].
- Further semantic notions can be modelled with coalgebras by changing the base category from $\mathcal{C} = \mathsf{Set}$ to the Eilenberg-Moore [33] or Kleisli category [20] of a monad, to nominal sets [26, 28], or to partially ordered sets [6].

The category of coalgebras inherits many properties from the base-category $\mathcal{C}$. For instance, we have the following standard result:

▶ **Corollary 2.6.** *The forgetful functor $\mathsf{Coalg}(F) \to \mathcal{C}$ creates all colimits. That is, the colimit of a diagram $D \colon \mathcal{D} \to \mathsf{Coalg}(F)$ exists, if $U \cdot D \colon \mathcal{D} \to \mathcal{C}$ has a colimit, and moreover, there is a unique coalgebra structure on $\operatorname{colim}(UD)$ making it the colimit of D and making the colimit injections of $\operatorname{colim} D$ coalgebra morphisms.*

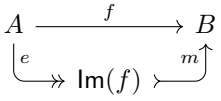


(a) F-Coalgebras.

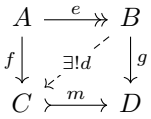


(b) F-Algebras.

Figure 1 A homomorphism between ....



(a) Factorization.



(b) Diagonal fill-in.

Figure 2 Diagrams for Definition 2.10.

On the other hand, we do not necessarily have all limits in $\mathsf{Coalg}(F)$. If F preserves a limit of a diagram $D\colon \mathcal{D} \to \mathcal{C}$, then the limit also exists in $\mathsf{Coalg}(F)$.

Coalgebras model systems with a transition structure, and pointed coalgebras extend this by a notion of initial state:

▶ **Definition 2.7.** *For an object $I \in \mathcal{C}$, an I-*pointed *F-coalgebra (C, c, i_C) is an F-coalgebra (C, c) together with a morphism $i_C\colon I \to C$. A pointed coalgebra morphism $h\colon (C, c, i_C) \to (D, d, i_D)$ is a coalgebra morphism $h\colon (C, c) \to (D, d)$ that preserves the point: $i_D = h \cdot i_C$.*

The category of I-pointed F-coalgebras is denoted by $\mathsf{Coalg}_I(F)$.

▶ **Example 2.8.** For $I := 1$ in Set, a pointed coalgebra (C, c, i_C) for $FX = 2 \times X^A$ is a deterministic automaton, where the initial state is given by the map $i_C\colon 1 \to C$.

The point can also be understood as an algebraic flavour. In general, coalgebras are dual to F-algebras in the following sense.

▶ **Definition 2.9.** *An F-algebra (for a functor $F\colon \mathcal{C} \to \mathcal{C}$) is a morphism $a\colon FA \to A$, an algebra homomorphism $h\colon (A, a) \to (B, b)$ is a morphism $h\colon A \to B$ fulfilling $b \cdot Fh = h \cdot a$ (Figure 1b). The category of F-algebras is denoted by $\mathsf{Alg}(F)$.*

In other words, $\mathsf{Alg}(F) = \mathsf{Coalg}(F^{\mathsf{op}})^{\mathsf{op}}$ for $F^{\mathsf{op}}\colon \mathcal{C}^{\mathsf{op}} \to \mathcal{C}^{\mathsf{op}}$. The I-pointed coalgebras thus are also algebras for the constant I functor. Most of the results of the present paper also apply to algebras for a functor $F\colon \mathcal{C} \to \mathcal{C}$.

2.2 Factorization Systems

The process of minimizing a system constructs a quotient or subobject of the state space, where the notions of quotient and subobject respectively stem from a factorization system in the category of interest. This generalizes the well-known image factorization of a map into a surjective and an injective map:

▶ **Definition 2.10** [2, Definition 14.1]**.** *Given classes of morphisms $\mathcal{E}$ and $\mathcal{M}$ in $\mathcal{C}$, we say that $\mathcal{C}$ has an $(\mathcal{E}, \mathcal{M})$-*factorization system *provided that:*

1. *$\mathcal{E}$ and $\mathcal{M}$ are closed under composition with isomorphisms.*
2. *Every morphism $f\colon A \to B$ in $\mathcal{C}$ has a factorization $f = m \cdot e$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$ (Figure 2a). We write $\mathsf{Im}(f)$ for the intermediate object, $\twoheadrightarrow$ for morphisms $e \in \mathcal{E}$, and $\rightarrowtail$ for morphisms $m \in \mathcal{M}$.*
3. *For each commutative square $g \cdot e = m \cdot f$ with $m \in \mathcal{M}$ and $e \in \mathcal{E}$, there exists a unique diagonal fill-in d with $m \cdot d = g$ and $d \cdot e = f$ (Figure 2b).*

▶ **Example 2.11.** In Set, we have an $(\mathsf{Epi}, \mathsf{Mono})$-factorization system where Epi is the class of surjective maps, and Mono the class of injective maps. The image of a map $f\colon A \to B$ is given by

$$\mathsf{Im}(f) = \{b \in B \mid \text{there exists } a \in A \text{ with } f(a) = b\}.$$

canonically yielding maps $e\colon A \twoheadrightarrow \mathsf{Im}(f)$ and $m\colon \mathsf{Im}(f) \rightarrowtail B$. Note that one can also regard $\mathsf{Im}(f)$ as a set of equivalence classes of A:

$$\mathsf{Im}(f) \cong \big\{\{a' \in A \mid f(a') = f(a)\} \mid a \in A\big\}.$$

Intuitively, the diagonal fill-in property (Definition 2.10.3, also called *diagonal lifting*) provides a way of defining a map d on equivalence classes (given by the surjective map at the top) and with a restricted codomain (given by the injective map at the bottom).

▶ **Example 2.12.** In general, the elements of $\mathcal{E}$ are not necessarily epimorphisms and the elements of $\mathcal{M}$ are not necessarily monomorphisms. In particular, every category has an $(\mathcal{E}, \mathcal{M})$-factorization system with $\mathcal{E} := \mathsf{Iso}$ being the class of isomorphisms and $\mathcal{M} := \mathsf{Mor}$ being the class of all morphisms (and also vice-versa).

▶ **Definition 2.13.** *An $(\mathcal{E}, \mathcal{M})$-factorization system is called* proper *if $\mathcal{E} \subseteq \mathsf{Epi}$ and $\mathcal{M} \subseteq \mathsf{Mono}$.*

These two conditions of properness are independent. In fact, $\mathcal{M} \subseteq \mathsf{Mono}$ is equivalent to every split-epimorphism being in $\mathcal{E}$ [2, Prop. 14.11]. In the literature, it is often required that the factorization system is proper, and in fact a proper factorization system arises in complete or cocomplete categories:

▶ **Example 2.14.** Every complete category has a $(\mathsf{StrongEpi}, \mathsf{Mono})$-factorization system [2, Thm. 14.17 and 14C(d)] and also an $(\mathsf{Epi}, \mathsf{StrongMono})$-factorization system [2, Thm. 14.19, and 14C(f)]. By duality, every cocomplete category has so as well.

▶ Remark 2.15. $(\mathcal{E}, \mathcal{M})$-factorization systems have many properties known from surjective and injective maps on Set [2, Chp. 14]:

1. $\mathcal{E} \cap \mathcal{M}$ is the class of isomorphisms of $\mathcal{C}$.
2. If $f \cdot g \in \mathcal{M}$ and $f \in \mathcal{M}$, then $g \in \mathcal{M}$. If $\mathcal{M} \subseteq \mathsf{Mono}$, then $f \cdot g \in \mathcal{M}$ implies $g \in \mathcal{M}$.
3. $\mathcal{E}$ and $\mathcal{M}$ are respectively closed under composition.
4. $\mathcal{M}$ is stable under pullbacks, $\mathcal{E}$ is stable under pushouts.

The stability generalizes as follows to wide pullbacks and pushouts:

▶ **Lemma 2.16.** *$\mathcal{M}$ is stable under wide pullbacks: for a family $(f_i\colon A_i \to B)_{i\in I}$ and its wide pullback $(\mathsf{pr}_i\colon P \to A_i)_{i\in I}$, a projection $\mathsf{pr}_j\colon P \to A_j$ is in $\mathcal{M}$ if f_i is in $\mathcal{M}$ for all $i \in I \setminus \{j\}$.*

A factorization system also provides notions of subobjects and quotients, generalizing the notions of subset and quotient sets:

▶ **Definition 2.17.** *For a class $\mathcal{M}$ of morphisms, an* $\mathcal{M}$-subobject *of an object X is a pair (S, s) where $s\colon S \rightarrowtail X$ is in $\mathcal{M}$. Two $\mathcal{M}$-subobjects (s, S), (s', S') are called* isomorphic *if there is an isomorphism $\phi\colon S \to S'$ with $\phi \cdot s = s'$. We write $(s, S) \leq (s', S')$ if there is a morphism $h\colon S \to S'$ with $s' \cdot h = s$. Dually, an* $\mathcal{E}$-quotient *of X is pair (Q, q) for a morphism $q\colon X \twoheadrightarrow Q$ $(q \in \mathcal{E})$. If $(\mathcal{E}, \mathcal{M})$ is fixed from the context, we simply speak of subobjects and quotients.*

For subobjects, it is often required that $\mathcal{M}$ is a class of monomorphisms [2, Def. 7.77], but many of the results in the present work hold without this assumption. If $\mathcal{M}$ is so, then the subobjects of a given object X form a preordered class. Moreover, the subobjects form a preordered set iff $\mathcal{C}$ is $\mathcal{M}$-wellpowered. This is in fact the definition: $\mathcal{C}$ is $\mathcal{M}$-wellpowered if for

each $X \in \mathcal{C}$ there is (up to isomorphism) only a set of $\mathcal{M}$-subobjects. On Set, the isomorphism classes of (Mono-)subobjects of X correspond to subsets of X and the isomorphism classes of (Epi-)quotients of X correspond to partitions on X.

If $(\mathcal{E}, \mathcal{M})$ forms a factorization system, then its axioms provide us with methods to construct and work with subobjects and quotients, e.g. the image factorization means that for every morphism, we obtain a quotient on its domain and a subobject of its codomain. The minimization of coalgebras amounts to the construction of certain subobjects or quotients with respect to a suitable factorization system in the category of coalgebras $\mathsf{Coalg}(F)$.

3 Factorization System for Coalgebras

If we have an $(\mathcal{E}, \mathcal{M})$-factorization system on the base category $\mathcal{C}$ on which we consider coalgebras for $F\colon \mathcal{C} \to \mathcal{C}$, then it is natural to consider coalgebra morphisms whose underlying $\mathcal{C}$-morphism is in $\mathcal{E}$, resp. $\mathcal{M}$:

▶ **Definition 3.1.** *Given a class of $\mathcal{C}$-morphisms $\mathcal{E}$, we say that an F-coalgebra morphism $h\colon (C, c) \to (D, d)$ is* $\mathcal{E}$-carried *if $h\colon C \to D$ is in $\mathcal{E}$.*

This induces the standard notions of subcoalgebra and quotient coalgebras as instances of $\mathcal{M}$-subobjects and $\mathcal{E}$-quotients in $\mathsf{Coalg}(F)$: an $\mathcal{M}$-subcoalgebra of (C, c) is an ($\mathcal{M}$-carried)-subobject of (C, c) (in $\mathsf{Coalg}(F)$), i.e. is represented by an $\mathcal{M}$-carried homomorphism $m\colon (S, s) \rightarrowtail (C, c)$. Likewise, a *quotient* of a coalgebra (C, c) is an ($\mathcal{E}$-carried)-quotient of (C, c) (in $\mathsf{Coalg}(F)$), i.e. is represented by a coalgebra morphism $e\colon (C, c) \twoheadrightarrow (Q, q)$ carried by an epimorphism.

Note that for the case where $\mathcal{M}$ is the class of monomorphisms, the monomorphisms in $\mathsf{Coalg}(F)$ coincide with the Mono-carried homomorphisms only under additional assumptions:

▶ **Lemma 3.2.** *If weak kernel pairs exist in $\mathcal{C}$ and are preserved by $F\colon \mathcal{C} \to \mathcal{C}$, then the monomorphisms in $\mathsf{Coalg}(F)$ are precisely the Mono-carried coalgebra homomorphisms.*

This is a commonly known criterion, and Gumm and Schröder [19, Example 3.5] present a functor not preserving kernel pairs and a monic coalgebra homomorphism that is not carried by a monomorphism.

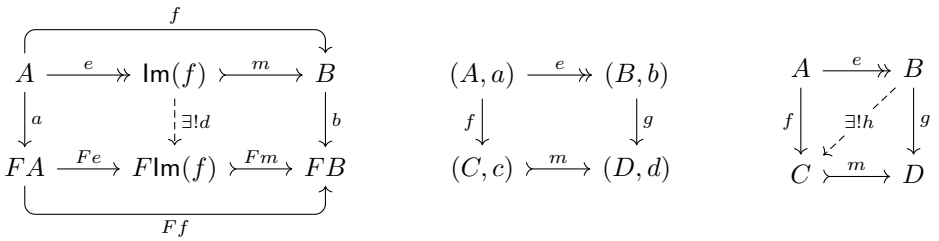
For the construction of quotient coalgebras and subcoalgebras, it is handy to have the factorization system directly in $\mathsf{Coalg}(F)$. It is a standard result that the image factorization of homomorphisms lifts (see e.g. [27, Lemma 2.5]). Under assumptions on $\mathcal{E}$ and $\mathcal{M}$, Kurz shows that the factorization system lifts to Coalg(F) [25, Theorem 1.3.7] (and to other categories with a forgetful functor to the base category $\mathcal{C}$).

In fact, the factorization system always lifts to $\mathsf{Coalg}(F)$ under the condition that F preserves $\mathcal{M}$. By this condition, we mean that $m \in \mathcal{M}$ implies $Fm \in \mathcal{M}$.

▶ **Lemma 3.3.** *If $F\colon \mathcal{C} \to \mathcal{C}$ preserves $\mathcal{M}$, then the $(\mathcal{E}, \mathcal{M})$-factorization system lifts from $\mathcal{C}$ to an ($\mathcal{E}$-carried, $\mathcal{M}$-carried)-factorization system in $\mathsf{Coalg}(F)$. The factorization of an F-coalgebra homomorphism $h\colon (C, c) \to (D, d)$ is given by that of the underlying morphism $h\colon C \to D$.*

Proof. We verify Definition 2.10:

1. The $\mathcal{E}$- and $\mathcal{M}$-carried morphisms are closed under composition with isomorphisms, respectively.
2. Given an F-coalgebra morphism $f\colon (A, a) \to (B, b)$, consider its factorization $f = m \cdot e$ in $\mathcal{C}$. Since F preserves $\mathcal{M}$, we have $Fm \in \mathcal{M}$ and thus can apply the diagonal fill-in

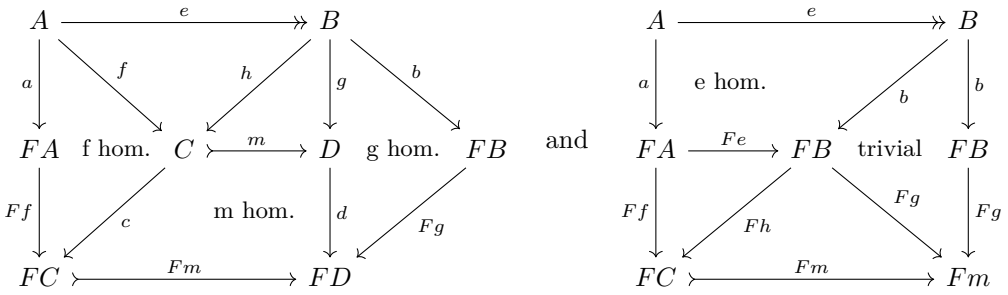


(a) Factorization. **(b)** Premise in $\mathsf{Coalg}(F)$. **(c)** Fill-in in $\mathcal{C}$.

Figure 3 Diagrams for the proof of Lemma 3.3.

property (Definition 2.10.3) to the coalgebra morphism square of f (Figure 3a). This defines a unique coalgebra structure d on $\mathsf{Im}(f)$ making e and m coalgebra morphisms.

3. In order to check the diagonal-lifting property of the ($\mathcal{E}$-carried, $\mathcal{M}$-carried)-factorization system, consider a commutative square $g \cdot e = m \cdot f$ in $\mathsf{Coalg}(F)$ with $m \in \mathcal{M}$, $e \in \mathcal{E}$ as depicted in Figure 3b. In $\mathcal{C}$, there exists a unique $h\colon B \to C$ with $h \cdot e = f$ and $m \cdot h = g$ (Figure 3c). We only need to prove that $h\colon B \to C$ is a coalgebra homomorphism $(B, b) \to (C, c)$, i.e. that $c \cdot h = Fh \cdot b$. We prove this equality by showing that both $c \cdot h$ and $Fh \cdot b$ are diagonals in a commutative square of the form of Definition 2.10.3. Indeed, we have the commutative squares:



By the uniqueness of the diagonal in Definition 2.10.3, $c \cdot h = Fh \cdot b$. ◀

▶ Remark 3.4. The condition that F preserves $\mathcal{M}$ is commonly met. For Set and $\mathcal{M}$ being the class of injective maps, it can be assumed wlog for coalgebraic purposes that F preserves injective maps: every set functor preserves injective maps with non-empty domain and only needs to be modified on $\emptyset$ in order to preserve all injective maps [32]. The resulting functor has an isomorphic category of coalgebras.

We have the dual result for F-algebras:

▶ **Lemma 3.5.** *If $F\colon \mathcal{C} \to \mathcal{C}$ preserves $\mathcal{E}$, then the $(\mathcal{E}, \mathcal{M})$-factorization system lifts from $\mathcal{C}$ to $\mathsf{Alg}(F)$.*

Proof. We have an $(\mathcal{M}, \mathcal{E})$-factorization system in $\mathcal{C}^{\mathsf{op}}$. By Lemma 3.3, this factorization system lifts to $\mathsf{Coalg}(F^{\mathsf{op}})$ since $F^{\mathsf{op}}\colon \mathcal{C}^{\mathsf{op}} \to \mathcal{C}^{\mathsf{op}}$ preserves $\mathcal{E}$. Thus, we have an ($\mathcal{E}$-carried, $\mathcal{M}$-carried)-factorization system in $\mathsf{Alg}(F) = \mathsf{Coalg}(F^{\mathsf{op}})^{\mathsf{op}}$. ◀

This lifting result even holds for Eilenberg-Moore algebras for a monad $T\colon \mathcal{C} \to \mathcal{C}$. The Eilenberg-Moore category of a monad T is a full subcategory of $\mathsf{Alg}(T)$ containing those algebras that interact coherently with the structure of the monad T, see e.g. Awodey [5] for details:

▶ **Lemma 3.6.** *If a monad $T\colon \mathcal{C} \to \mathcal{C}$ preserves $\mathcal{E}$, then the $(\mathcal{E}, \mathcal{M})$-factorization system lifts from $\mathcal{C}$ to the Eilenberg-Moore category of T.*

The factorization system lifts further also to pointed coalgebras:

▶ **Lemma 3.7.** *If $F\colon \mathcal{C} \to \mathcal{C}$ preserves $\mathcal{M}$, then the $(\mathcal{E}, \mathcal{M})$-factorization system lifts from $\mathcal{C}$ to $\mathsf{Coalg}_I(F)$.*

Proof. A combination of Lemma 3.3 and 3.5, using that the constant functor preserves $\mathcal{E}$-morphisms. ◀

4 Minimality in a Category

Having seen multiple categories with an $(\mathcal{E}, \mathcal{M})$-factorization system, we can now define the minimality of objects abstractly.

▶ **Definition 4.1.** *Given a category $\mathcal{K}$ with an $(\mathcal{E}, \mathcal{M})$-factorization system, an object C of $\mathcal{K}$ is called* $\mathcal{M}$-minimal *if every morphism $h\colon D \rightarrowtail C$ in $\mathcal{M}$ is an isomorphism.*

▶ Remark 4.2. Every $(\mathcal{E}, \mathcal{M})$-factorization system on $\mathcal{K}$ is an $(\mathcal{M}, \mathcal{E})$-factorization system on $\mathcal{K}^{\mathsf{op}}$, and thus induces a dual notion of $\mathcal{E}$-minimality: an object C of $\mathcal{K}$ is called *$\mathcal{E}$-minimal* if every $h\colon C \twoheadrightarrow D$ in $\mathcal{E}$ is an isomorphism.

In the following, $\mathcal{K}$ will denote the category in which we consider the minimal objects, e.g. a category of coalgebras for a functor $F\colon \mathcal{C} \to \mathcal{C}$.

▶ **Assumption 4.3.** In the following, assume that the category $\mathcal{K}$ has an $(\mathcal{E}, \mathcal{M})$-factorization system. Whenever we consider a category of coalgebras for a functor $F\colon \mathcal{C} \to \mathcal{C}$ in the following, we achieve this by assuming that $\mathcal{C}$ has an $(\mathcal{E}, \mathcal{M})$-factorization system and that F preserves $\mathcal{M}$.

The leading examples of the minimality notion in the present work are the following two instances in coalgebras:

▶ **Instance 4.4.** For $\mathcal{K} := \mathsf{Coalg}_I(F)$, the ($\mathcal{M}$-carried-)minimal objects are the *reachable* coalgebras, as introduced by Adámek et al. [4]. Concretely, an I-pointed F-coalgebra (C, c, i_C) is reachable if it has no (proper) pointed subcoalgebra, equivalently, if every $\mathcal{M}$-carried coalgebra morphism $m\colon (S, s, i_S) \rightarrowtail (C, c, i_C)$ is necessarily an isomorphism [4].

In Set, this corresponds to the usual notion of reachability: if a state $x \in C$ is contained in a subcoalgebra $m\colon (S, s, i_S) \rightarrowtail (C, c, i_C)$, then all successors of x need to be contained in the subcoalgebra as well, since m is a coalgebra homomorphism. Moreover, the subcoalgebra has to contain the point $i_C\colon I \to C$, and thus also all its successors, and in total all states reachable from i_C in finitely many steps. Hence, (C, c, i_C) is reachable if it is not possible to omit any state in a pointed subcoalgebra (S, s, i_S), i.e. if any such injective m is a bijection.

▶ **Instance 4.5.** For $\mathcal{K} := \mathsf{Coalg}(F)^{\mathsf{op}}$, the ($\mathcal{E}$-carried-)minimal objects are called *simple* coalgebras, as mentioned by Gumm [22]. Usually, a simple coalgebra is defined as a coalgebra that does not have any proper quotient [36].[1]

[1] Gumm [22, p. 34] defines a simple coalgebra as the quotient of a coalgebra on Set modulo behavioural equivalence.

X is $\mathcal{M}$-minimal	X is $\mathcal{E}$-minimal
$\Leftrightarrow$ every $Y \rightarrowtail X$ is an isomorphism	$\Leftrightarrow$ every $X \twoheadrightarrow Y$ is an isomorphism
$\Leftrightarrow$ every $Y \to X$ is in $\mathcal{E}$	$\Leftrightarrow$ every $X \to Y$ is in $\mathcal{M}$
if $\mathcal{E} = \mathsf{Epi}$ and $\mathcal{K}$ has weak equalizers: $\Leftrightarrow$ all parallel $X \rightrightarrows Y$ equal	if $\mathcal{M} = \mathsf{Mono}$ and $\mathcal{K}$ has weak coequalizers: $\Leftrightarrow$ all parallel $Y \rightrightarrows X$ equal ('X subterminal')

Figure 4 Equivalent characterizations of $\mathcal{M}$-minimality and $\mathcal{E}$-minimality in a category $\mathcal{K}$.

In Set, a coalgebra is simple iff all states have different behaviour – this characterization follows directly the following equivalent characterization of minimal objects as we will see in Instance 4.7:

▶ **Lemma 4.6.** *An object C in $\mathcal{K}$ is $\mathcal{M}$-minimal iff every $h\colon D \to C$ is in $\mathcal{E}$.*

Proof. In the 'if' direction, consider some $\mathcal{M}$-morphism $h\colon D \to C$. By the assumption, h is also in $\mathcal{E}$ and thus an isomorphism. In the 'only if' direction, take some morphism $h\colon D \to C$ and consider its $(\mathcal{E},\mathcal{M})$-factorization $e\colon D \twoheadrightarrow \mathsf{Im}(h)$ and $m\colon \mathsf{Im}(h) \rightarrowtail C$ with $h = m \cdot e$. Since C is $\mathcal{M}$-minimal, m is an isomorphism and thus $h = m \cdot e$ is in $\mathcal{E}$. ◀

▶ **Instance 4.7.** For $\mathcal{K} := \mathsf{Coalg}(F)^{\mathsf{op}}$, an F-coalgebra (C, c) is simple iff every F-coalgebra homomorphism $h\colon (C, c) \to (D, d)$ is $\mathcal{M}$-carried.

In Set, this equivalence shows that the simple coalgebras are precisely those coalgebras for which behavioural equivalence coincides with behavioural equivalence:

- If states $x, y \in C$ are behaviourally equivalent, then there is some $h\colon (C, c) \to (D, d)$ with $h(x) = h(y)$. By Instance 4.7, h must be injective and thus $x = y$.
- Conversely, if all states in (C, c) have different behaviour, then every $h\colon (C, c) \to (D, d)$ is necessarily injective by Definition 2.4. Thus, by Instance 4.7, (C, c) is simple.

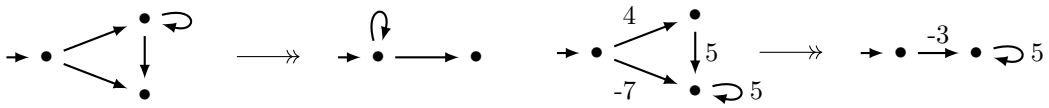
Gumm already noted that in Set, every outgoing coalgebra morphism from a simple coalgebra is injective [22, Hilfssatz 3.6.3] – but it was not used to characterize simplicity. If $\mathcal{E}$, resp. $\mathcal{M}$, happens to be the class of epimorphisms, resp. monomorphisms, another characterization of minimality exists:

▶ **Lemma 4.8.** *Assume $\mathcal{E} = \mathsf{Epi}$ and weak equalizers in $\mathcal{K}$, then X is $\mathcal{M}$-minimal iff there is at most one morphism $u\colon C \to D$ for every $D \in \mathcal{K}$.*

Dually, given $\mathcal{M} = \mathsf{Mono}$ and weak coequalizers, C is $\mathcal{E}$-minimal iff C is subterminal*, that is, iff there is a most one $u\colon D \to C$ for every $D \in \mathcal{K}$.*

The name *subterminal* stems from the fact that if $\mathcal{K}$ has a terminal object, its subobjects are the subterminal objects.

▶ **Instance 4.9.** For $\mathcal{K} := \mathsf{Coalg}(F)^{\mathsf{op}}$, assume $\mathcal{M} = \mathsf{Mono}$ and that F preserves weak kernel pairs and that the base category $\mathcal{C}$ has coequalizers. Hence, the monomorphisms in $\mathsf{Coalg}(F)$ are precisely the Mono-carried homomorphisms (Lemma 3.2) and the assumption of Lemma 4.8 is met. Consequently, the simple coalgebras are precisely the *subterminal* coalgebras. If the final coalgebra exists, then its subcoalgebras are precisely the simple coalgebras. For a non-example, Gumm and Schröder [19, Example 3.5] provide a functor not preserving weak kernel pairs and a subterminal coalgebra that is not simple.



(a) For $FX = \mathcal{P}X$ **(b)** For $FX = (\mathbb{R}, +, 0)^{(X)}$

Figure 5 Examples of simple quotients in F-coalgebras.

We have now established a series of equivalent characterizations of minimality (Figure 4) and will now discuss how to construct minimal objects. This process of minimization – i.e. of constructing the reachable part or the simple quotient of a coalgebra – is abstracted as follows:

▶ **Definition 4.10.** *An* $\mathcal{M}$-minimization *of* $C \in \mathcal{K}$ *is a morphism* $m\colon D \rightarrowtail C$ *in* $\mathcal{M}$ *where* D *is* $\mathcal{M}$*-minimal.*

In fact, we will show that an $\mathcal{M}$-minimization is unique, so we can speak of *the* $\mathcal{M}$-minimization.

▶ **Instance 4.11.** The task of finding an $\mathcal{M}$-minimization of a given $C \in \mathcal{K}$ instantiates to the standard minimization tasks on coalgebras:

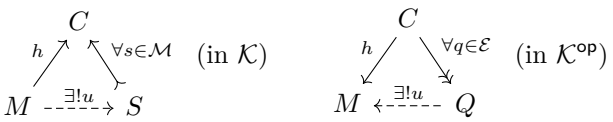
- For $\mathcal{K} := \mathsf{Coalg}_I(F)$, an $\mathcal{M}$-minimization of a given pointed coalgebra (C, c, i_C) is called its *reachable subcoalgebra* [4]. This is a subcoalgebra obtained by removing all unreachable states. The explicit definition is: the reachable subcoalgebra of (C, c, i_C) is a (pointed) subcoalgebra $h\colon (R, r, i_R) \rightarrowtail (C, c, i_C)$ where (R, r, i_R) itself has no proper (pointed) subcoalgebras.
- For $\mathcal{K} := \mathsf{Coalg}(F)^{\mathsf{op}}$, an $\mathcal{E}$-minimization of a given coalgebra (C, c) is called the *simple quotient* of (C, c) [22]. The explicit definition is: the simple quotient of (C, c) is a quotient $h\colon (C, c) \twoheadrightarrow (Q, q)$ where (Q, q) itself has no proper quotient coalgebra.
  In Set, this is a quotient in which all behaviourally equivalent states are identified, in other words, the simple quotient of (C, c) is the unique coalgebra structure on $C/\!\sim$. Examples of simple quotients can be found in Figure 5. Since all states in the codomain of the surjective homomorphisms are behaviourally inequivalent, the respective codomains are simple.

▶ **Example 4.12.** For the trivial factorization systems (Example 2.12), we have:

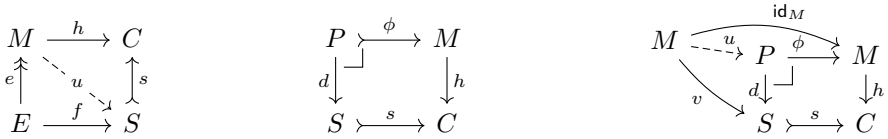
- For the $(\mathsf{Iso}, \mathsf{Mor})$-factorization system, the Iso-minimization of an object X is X itself.
- For the $(\mathsf{Mor}, \mathsf{Iso})$-factorization system on category, if a strict initial object 0 exist, then it is the Mor-minimization of every $X \in \mathcal{C}$. Recall that an initial object 0 is called *strict* if every morphism with codomain 0 is an isomorphism.

It is well-defined to speak of *the* $\mathcal{M}$-minimization of an object C, because it is unique:

▶ **Lemma 4.13.** *Consider* $h\colon M \to C$ *with* $\mathcal{M}$*-minimal* M *and an* $\mathcal{M}$*-subobject* $s\colon S \rightarrowtail C$*. The pullback of* s *along* h *exists iff* h *factors uniquely through* s*, that is, iff there is a unique* $u\colon M \to S$ *with* $s \cdot u = h$*.*



Proof. In the "if"-direction, let $d\colon M \to S$ be the unique morphism with $s \cdot d = h$. The pullback is simply given by M itself with projections $\mathsf{id}_M\colon M \to M$ and $d\colon M \to S$. To verify its universal property, consider $e\colon E \to M$, $f\colon E \to S$ with $h \cdot e = s \cdot f$ (Figure 6a). Since M is $\mathcal{M}$-minimal, $e\colon E \to M$ is in $\mathcal{E}$ (Lemma 4.6). Thus, we can apply the diagonal lifting property to $h \cdot e = s \cdot f$ yielding a diagonal u with $s \cdot u = h$ and $u \cdot e = f$. Thus, $d = u$ and $d \cdot e = f$, showing that $e\colon (E, e, f) \to (M, \mathsf{id}_M, d)$ is the mediating cone morphism. Its uniqueness is clear because id_M is isomorphic.



(a) "If"-direction. **(b)** Premise of "only if". **(c)** Uniqueness for "only if".

Figure 6 Diagrams for the proof of Lemma 4.13.

In the 'only if'-direction, consider the pullback (P, ϕ, d) (Figure 6b). Since $\mathcal{M}$-morphisms are stable under pullback (Remark 2.15.4), ϕ is in $\mathcal{M}$, too. By the minimality of M, the $\mathcal{M}$-morphism ϕ is an isomorphism and we have $d \cdot \phi^{-1}\colon M \to S$.

In order to see that $d \cdot \phi^{-1}$ is indeed the unique morphism $M \to S$ making the triangle commute, consider an arbitrary $v\colon M \to S$ with $s \cdot v = h = h \cdot \mathsf{id}_M$. Thus, M is a competing cone for the pullback P and thus induces a morphism $u\colon M \to P$ with $d \cdot u = v$ and $\phi \cdot u = \mathsf{id}_M$ (Figure 6c). Since ϕ is an isomorphism, we have $u = \phi^{-1}$ and thus $v = d \cdot \phi^{-1}$ as desired. ◀

▶ **Corollary 4.14.** *If $\mathcal{K}$ has pullbacks of $\mathcal{M}$-morphisms along $\mathcal{M}$-morphisms and if there is an $\mathcal{M}$-minimization M of C, then M is the least $\mathcal{M}$-subobject of C and it is unique (up to unique isomorphism).*

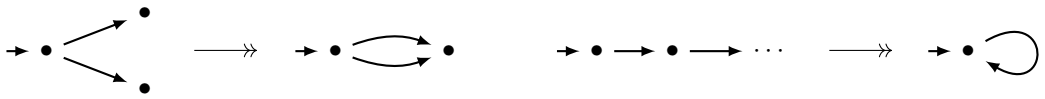
Proof. Consider Lemma 4.13 first for $h \in \mathcal{M}$ and then also with S being $\mathcal{M}$-minimal. ◀

▶ **Instance 4.15.** Not only the result but also the proof instantiates to the uniqueness results in the instances of reachable subcoalgebras and simple quotients:

1. If $\mathcal{C}$ has pullbacks of $\mathcal{M}$-morphisms (i.e. finite intersections) and $F\colon \mathcal{C} \to \mathcal{C}$ preserves them, then $\mathsf{Coalg}_I(F)$ has pullbacks of $\mathcal{M}$-carried homomorphisms. Given a reachable subcoalgebra (D, d, i_D) of (C, c, i_C), then it is the least I-pointed subcoalgebra of (C, c, i_C) (cf. [4, Notation 3.18]) and is unique up to isomorphism.
2. If $\mathcal{C}$ has pushouts of $\mathcal{E}$-morphisms, then $\mathsf{Coalg}(F)$ has pushouts of $\mathcal{E}$-carried homomorphisms. Hence, the simple quotient of a coalgebra (C, c) is the greatest quotient of (C, c) and unique up to isomorphism (e.g. [36, Lemma 2.9]).

There are instances where a minimization M exists, but where a mediating morphism in the sense of Lemma 4.13 is not unique:

▶ **Example 4.16** (Tree unravelling)**.** Let $\mathsf{Coalg}_I(F)_{\mathsf{reach}}$ be the category of reachable pointed F-coalgebras, i.e. the full subcategory $\mathsf{Coalg}_I(F)_{\mathsf{reach}} \subseteq \mathsf{Coalg}_I(F)$ such that $(C, c, i_C) \in \mathsf{Coalg}_I(F)_{\mathsf{reach}}$ iff it is reachable. For simplicity, restrict to $F\colon \mathsf{Set} \to \mathsf{Set}$ with the $(\mathsf{Epi}, \mathsf{Mono})$-factorization system. Thus, all morphisms in $\mathsf{Coalg}_I(F)_{\mathsf{reach}}$ are surjective (Lemma 4.6). Considering the (trivial) $(\mathsf{Mor}, \mathsf{Iso})$-factorization system on $\mathsf{Coalg}_I(F)_{\mathsf{reach}}$, a coalgebra (C, c, i_C) is (Mor-)minimal iff every coalgebra morphism $h\colon (D, d, i_D) \to (C, c, i_C)$ (with (D, d, i_D) also reachable) is an isomorphism. If (C, c, i_C) is Mor-minimal, then it is a tree: to see this, take h to be its tree unravelling (see e.g. Figure 7), and by the Mor-minimality, h is an isomorphism, so (C, c, i_C) is already a tree.



(a) Unravelling of siblings **(b)** Unravelling of a loop

Figure 7 Tree unravelling for $FX = \mathcal{B}X$.

This implies that if the (Mor-)minimization of a coalgebra exists, then it is its tree unravelling. For example, for $I = 1$ and the bag functor $FX = \mathcal{B}X$, we have the minimizations as illustrated in Figure 7. It is easy to see that for $FX = \mathcal{P}X$ however, no coalgebra (with at least one transition) has a Mor-minimization, because one can always duplicate successor states.[2]

For $FX = \mathcal{B}X$, all Mor-minimizations exist, but they are not unique up to unique isomorphism. Consider the tree unravelling $m\colon M \twoheadrightarrow X$ in Figure 7a. There is an isomorphism $\phi\colon M \to M$ that swaps the two successors of the initial state. Hence, $\phi \neq \mathsf{id}_B$, but $m \cdot \phi = m \cdot \mathsf{id}_M$, so M is unique up to isomorphism, but not unique up to unique isomorphism.

For proving the existence of an $\mathcal{M}$-minimization, we need to require that $\mathcal{M}$ is a subclass of the monomorphisms in $\mathcal{K}$. Under this assumption, we first establish the converse of Corollary 4.14:

▶ **Lemma 4.17.** *If $\mathcal{M} \subseteq \mathsf{Mono}$ and if the least $\mathcal{M}$-subobject M of X exists, then M is the $\mathcal{M}$-minimization of X.*

Proof. Let $m\colon M \rightarrowtail X$ be the least $\mathcal{M}$-subobject of X, and consider $s\colon S \rightarrowtail M$ in $\mathcal{M}$. Since $m \cdot s \in \mathcal{M}$, there is some $u\colon M \to S$ with $(m \cdot s) \cdot u = m$. Since m is monic, we obtain $s \cdot u = \mathsf{id}_M$. Hence, s is a split-epimorphism, and together with $s \in \mathcal{M} \subseteq \mathsf{Mono}$, s is an isomorphism. ◀

▶ **Proposition 4.18.** *If $\mathcal{M} \subseteq \mathsf{Mono}$, $\mathcal{K}$ has wide pullbacks of $\mathcal{M}$-morphisms, and is $\mathcal{M}$-wellpowered, then every object C of $\mathcal{K}$ has an $\mathcal{M}$-minimization.*

Proof. Since $\mathcal{K}$ is $\mathcal{M}$-wellpowered, all the $\mathcal{M}$-carried morphisms $m\colon M \rightarrowtail C$ form up to isomorphism a set S. The wide pullback of all $m \in S$ exists in $\mathcal{K}$ by assumption, denote it by $\mathsf{pr}_m\colon P \to M$ for $m\colon M \rightarrowtail C$. All $m' \in S$ are in $\mathcal{M}$ and so are all pr_m by Lemma 2.16. Hence, $p := m \cdot \mathsf{pr}_m\colon P \rightarrowtail C$ for an arbitrary $m \in S$ represents an $\mathcal{M}$-subobject, and moreover the least $\mathcal{M}$-subobject of C, as witnessed by the projections pr_m. By Lemma 4.17, P is the minimization of C. ◀

▶ **Instance 4.19.** This proof directly instantiates to the proofs of the existence of the reachable subcoalgebra and simple quotient:

1. In the reachability case, let $\mathcal{M}$ be a subclass of the monomorphisms, let the base category $\mathcal{C}$ have all $\mathcal{M}$-intersections, and let $F\colon \mathcal{C} \to \mathcal{C}$ preserve all intersections. Then the reachable part of a given pointed coalgebra (C, c, i_C) is obtained as the intersection of all pointed subcoalgebras of (C, c, i_C) [4].
   For $\mathcal{C} = \mathsf{Set}$, and $\mathcal{M}$ being the class of injective maps, all intersections exist. The condition that $F\colon \mathsf{Set} \to \mathsf{Set}$ preserves all intersections is mild: all finitary functors preserve all

[2] It can be conjectured that Mor-minimization of reachable F-coalgebras exist if F admits precise factorizations [37, Def. 3.1, 3.4].

intersections ([3, Proof of Lem. 8.8] or [38, Lem. 2.6.10]) and many non-finitary functors do as well, e.g. the powerset functor. An example of a functor that does not preserve all intersections is the filter functor [16, Sect. 5.3].

2. For the existence of simple quotients, let $\mathcal{E}$ be a subclass of the epimorphisms and let the base category $\mathcal{C}$ be cocomplete and $\mathcal{E}$-cowellpowered. Then every F-coalgebra (C,c) has a simple quotient given by the wide pushout of all quotient coalgebras ([4, Proposition 3.7], and [17] for $\mathcal{C} = \mathsf{Set}$).
   Every set has only a set of outgoing surjective maps, so all assumptions are met for $\mathcal{C} = \mathsf{Set}$, $\mathcal{E}$ the surjective maps, and every Set-functor F.

▶ Remark 4.20. All observations on simple quotients also apply to pointed coalgebras: An I-pointed F-coalgebra is simple iff it is $\mathcal{E}$-carried-minimal in $\mathcal{K} := \mathsf{Coalg}_I(F)^{\mathsf{op}}$. The forgetful functor

$$\mathsf{Coalg}_I(F) \longrightarrow \mathsf{Coalg}(F)$$

preserves and reflects simple coalgebras and simple quotients (note that for every pointed coalgebra (C, c, i_C), the slice categories $(C, c, i_C)/\mathsf{Coalg}_I(F)$ and $(C, c)/\mathsf{Coalg}(F)$ are isomorphic). For the sake of simplicity, we will not state the results explicitly for simple coalgebras in $\mathsf{Coalg}_I(F)$.

▶ **Definition 4.21.** *We denote by $J\colon \mathcal{K}_{\min} \hookrightarrow \mathcal{K}$ the full subcategory formed by the $\mathcal{M}$-minimal objects of $\mathcal{K}$.*

In the existence proof of minimal objects (Proposition 4.18) we only required (wide) pullbacks where all morphisms in the diagram are in $\mathcal{M}$. We obtain additional properties if we assume the pullback along $\mathcal{M}$-morphisms, i.e. pullbacks where only one of the two morphisms is in $\mathcal{M}$:

▶ **Proposition 4.22.** *Suppose that pullbacks along $\mathcal{M}$-morphisms exist in $\mathcal{K}$ and that every object of $\mathcal{K}$ has an $\mathcal{M}$-minimization. Then $J\colon \mathcal{K}_{\min} \hookrightarrow \mathcal{K}$ is a coreflective subcategory. Its right-adjoint $R\colon \mathcal{K} \to \mathcal{K}_{\min}$ $(J \dashv R)$ sends an object to its $\mathcal{M}$-minimization; in particular, minimization is functorial.*

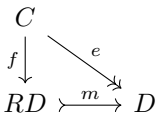
Proof. The universal property of J follows directly from Lemma 4.13: To this end, it suffices to consider R as an object assignment. Given a morphism $h\colon M \to X$ where M is $\mathcal{M}$-minimal, we need to show that it factorizes uniquely through the $\mathcal{M}$-minimization $s\colon S \rightarrowtail X$ of D, that is $RD := S$. Since the pullback of h along s exists by assumption, Lemma 4.13 yields us the desired unique factorization $u\colon M \to S$ with $s \cdot u = h$. ◀

▶ **Instance 4.23.** For both of our main instances, this adjunction has been observed before:

1. If $F\colon \mathcal{C} \to \mathcal{C}$ preserves inverse images (w.r.t. $\mathcal{M}$), then pullbacks along $\mathcal{M}$-carried homomorphisms exist in $\mathsf{Coalg}_I(F)$. Hence, the reachable I-pointed F-coalgebras form a coreflective subcategory of $\mathsf{Coalg}_I(F)$, where the coreflector maps a pointed coalgebra to its reachable part [39, Thm 5.23]
2. The simple coalgebras form a reflective subcategory of $\mathsf{Coalg}(F)$, and the reflector sends a coalgebra to its simple quotient, under the assumption that the base category has pushouts along $\mathcal{E}$-morphisms. For coalgebras in Set, the adjunction $J \vdash R$ has been shown by Gumm [17, Theorem 2.3].

▶ **Corollary 4.24.** *If pullbacks along $\mathcal{M}$-morphisms exist in $\mathcal{K}$ and all $\mathcal{M}$-minimizations exist, then $\mathcal{M}$-minimal objects are closed under $\mathcal{E}$-quotients.*

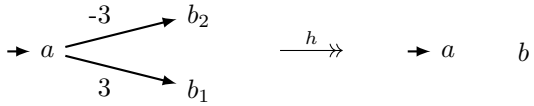
Proof. Consider an $\mathcal{E}$-morphism $e\colon C \twoheadrightarrow D$ where C is $\mathcal{M}$-minimal. Take the adjoint transpose $f\colon C \to RD$ with $m \cdot f = e$ where $m\colon RD \rightarrowtail D$ is the $\mathcal{M}$-minimization of D:



Since RD is $\mathcal{M}$-minimal, f is in $\mathcal{E}$ (Lemma 4.6). Moreover, $f \in \mathcal{E}$ and $m \cdot f \in \mathcal{E}$ imply $m \in \mathcal{E}$ (Remark 2.15.2), hence $m \in \mathcal{E} \cap \mathcal{M}$ is an isomorphism. ◀

▶ **Example 4.25.**

1. If F preserves inverse images, then reachable F-coalgebras are closed under quotients [39, Cor. 5.24]. Note that if F does not preserve inverse images, then a quotient of a reachable F-coalgebra may not be reachable. For example, in (pointed) coalgebras for the monoid-valued functor $(\mathbb{R}, +, 0)^{(-)}$ there is the coalgebra quotient with $h(b_1) = h(b_2) = b$:



   Since transition weights may cancel out each other ($-3 + 3 = 0$), the codomain of h is not reachable even though its domain is.
2. If the base category $\mathcal{C}$ has pushouts along $\mathcal{E}$-morphisms, then simple F-coalgebras are closed under subcoalgebras. For $\mathcal{C} = \mathsf{Set}$, this is obvious: if in a coalgebra (C, c), all states are of pairwise different behaviour, then so they are in every subcoalgebra of (C, c).

5 Interplay of minimality notions

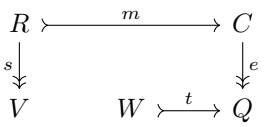
The two main aspects of minimization we have seen – reachability and minimization for observability – are closely connected on an abstract level and also interact well as we see in the following. In order to minimize a pointed coalgebra under both aspects, we have two options: first construct the reachable part and then the simple quotient, or we first form the simple quotient and then construct its reachable part. Given the existence of pullbacks of $\mathcal{M}$-morphisms along arbitrary morphisms, we can show that any order is fine.

In the abstract setting of a category $\mathcal{K}$ with an $(\mathcal{E}, \mathcal{M})$-factorization system we are transforming an object $C \in \mathcal{K}$ into an object C' that is $\mathcal{M}$-minimal in $\mathcal{K}$ and $\mathcal{E}$-minimal in $\mathcal{K}^{\mathsf{op}}$.

▶ **Proposition 5.1.** *Suppose $\mathcal{K}$ has an $(\mathcal{E}, \mathcal{M})$-factorization system such that all $\mathcal{M}$-minimizations in $\mathcal{K}$ and all $\mathcal{E}$-minimizations in $\mathcal{K}^{\mathsf{op}}$ exist. If $\mathcal{K}$ has pullbacks along $\mathcal{M}$-morphisms and pushouts along $\mathcal{E}$-morphisms, then for every C in $\mathcal{K}$ the following two constructions yield the same object:*

1. *The $\mathcal{M}$-minimization of C in $\mathcal{K}$ followed by its $\mathcal{E}$-minimization in $\mathcal{K}^{\mathsf{op}}$.*
2. *The $\mathcal{E}$-minimization of C in $\mathcal{K}^{\mathsf{op}}$ followed by its $\mathcal{M}$-minimization in $\mathcal{K}$.*

Proof. In the first approach, denote the $\mathcal{M}$-minimization of C by $m\colon R \rightarrowtail C$ and its $\mathcal{E}$-minimization by $s\colon R \twoheadrightarrow V$. In the other approach, denote the $\mathcal{E}$-minimization of C by $e\colon C \twoheadrightarrow Q$ and its $\mathcal{M}$-minimization by $t\colon W \rightarrowtail Q$:



We need to prove that V and W are isomorphic, making the above (then-closed) square commute. The $\mathcal{M}$-minimal objects form a coreflective subcategory (Proposition 4.22), so $e \cdot m$, whose domain is $\mathcal{M}$-minimal, factorizes through the $\mathcal{M}$-minimization of the codomain of $e \cdot m$, i.e. we have $h\colon R \to W$ with $t \cdot h = e \cdot m$. Since Q is $\mathcal{E}$-minimal, its $\mathcal{M}$-subobject W is also $\mathcal{E}$-minimal in $\mathcal{K}^{\mathsf{op}}$ (Corollary 4.24). The $\mathcal{E}$-minimal objects form a reflective subcategory (Proposition 4.22). Applying the reflection to $h\colon R \to W$, we obtain $\phi\colon V \to W$ with $h = \phi \cdot s$. Since V is $\mathcal{E}$-minimal (in $\mathcal{K}^{\mathsf{op}}$), ϕ is in $\mathcal{M}$, and since W is $\mathcal{M}$-minimal, ϕ is in $\mathcal{E}$, and thus ϕ is an isomorphism. ◀

In the concrete case of F-coalgebras, a coalgebra that is both simple and reachable is called a *well-pointed* coalgebra (see [4, Section 3.2]).

▶ **Instance 5.2.** If $F\colon \mathcal{C} \to \mathcal{C}$ fulfils all assumptions from the previous Instance 4.23 (and in particular preserves inverse images), then the construction of the simple quotient and the reachability construction for F-coalgebras can be performed in any order, yielding the same well-pointed coalgebra.

If F does not preserve inverse images, then in the construction of the simple quotient, transitions may cancel out each other and this may affect the reachability of states. We have seen an example for this in Example 4.25.1 where performing reachability first and observability second leads to a simple coalgebra in which states are unreachable, i.e. the result is not well-pointed. Hence, in contrast to the well-known automata minimization procedure, the minimization of a coalgebra in general has to be performed by first computing its simple quotient and secondly computing the reachable part in the simple quotient.

If F preserves inverse images, such as the functor for automata, any order is fine. In sets, the reachability computation is a simple breadth-first search [39], and hence runs in linear time. On the other hand, existing algorithms for computing the simple quotient for many Set-functors run in at least $n \cdot \log n$ time where n is the size of the coalgebra [15, 36]. Hence, the reachability analysis should be done first whenever possible.

6 Conclusions

We have seen a common ground for minimality notions in a category with various instances in a coalgebraic setting. The abstract results about the uniqueness and the existence of the minimization instantiate to the standard results for reachability and observability of coalgebras. Most of the general results even hold if the $(\mathcal{E}, \mathcal{M})$-factorization system is not proper. The tree unravelling of an automaton is an instance of minimization for a non-proper factorization system.

It remains for future work to relate the efficient algorithmic approaches to the minimization tasks: reachability is computed by breadth-first search [39, 7] and observability is computed by partition refinement algorithms [24, 36, 13]. Even though their run-time complexity differs – reachability is usually linear, whereas partition refinement algorithms are quasilinear or slower – they have striking similarities. All these algorithms compute a chain of subobjects resp. quotients on the carrier of the input coalgebra and terminate at the first element of the chain admitting a coalgebra structure compatible with the input coalgebra. It is thus likely that this relation can be made formal. A similar connection between the reachability of algebras and partition refinement on coalgebras is already known [30].

References

1 Peter Aczel and Nax Mendler. A final coalgebra theorem. In *Proc. Category Theory and Computer Science (CTCS)*, volume 389 of *Lecture Notes Comput. Sci.*, pages 357–365. Springer, 1989.

2 Jiří Adámek, Horst Herrlich, and George E. Strecker. *Abstract and Concrete Categories: The Joy of Cats*. Dover Publications, 2nd edition, 2009.

3 Jiří Adámek, Stefan Milius, and Lawrence S. Moss. Fixed points of functors. *Journal of Logical and Algebraic Methods in Programming*, 95:41–81, 2018.

4 Jiří Adámek, Stefan Milius, Lawrence S. Moss, and Lurdes Sousa. Well-pointed coalgebras. *Logical Methods in Computer Science*, 9(3:2):51 pp., 2013.

5 Steve Awodey. *Category Theory*. Oxford Logic Guides. OUP Oxford, 2010.

6 Adriana Balan and Alexander Kurz. Finitary functors: From set to preord and poset. In Andrea Corradini, Bartek Klin, and Corina Cîrstea, editors, *Algebra and Coalgebra in Computer Science - 4th International Conference, CALCO 2011, Winchester, UK, August 30 - September 2, 2011. Proceedings*, volume 6859 of *Lecture Notes in Computer Science*, pages 85–99. Springer, 2011. doi:10.1007/978-3-642-22944-2_7.

7 Simone Barlocco, Clemens Kupke, and Jurriaan Rot. Coalgebra learning via duality. In Mikolaj Bojanczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 62–79. Springer, 2019. doi:10.1007/978-3-030-17127-8_4.

8 Falk Bartels, Ana Sokolova, and Erik P. de Vink. A hierarchy of probabilistic system types. *Theor. Comput. Sci.*, 327(1-2):3–22, 2004. doi:10.1016/j.tcs.2004.07.019.

9 Nick Bezhanishvili, Clemens Kupke, and Prakash Panangaden. Minimization via duality. In C.-H. Luke Ong and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation - 19th International Workshop, WoLLIC 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings*, volume 7456 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2012. doi:10.1007/978-3-642-32621-9_14.

10 Michel Bidoit, Rolf Hennicker, and Alexander Kurz. On the duality between observability and reachability. In Furio Honsell and Marino Miculan, editors, *Foundations of Software Science and Computation Structures, 4th International Conference (FOSSACS 2001), Held as Part of ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings*, volume 2030 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2001. doi:10.1007/3-540-45315-6_5.

11 Filippo Bonchi, Marcello M. Bonsangue, Helle Hvid Hansen, Prakash Panangaden, Jan J. M. M. Rutten, and Alexandra Silva. Algebra-coalgebra duality in brzozowski's minimization algorithm. *ACM Trans. Comput. Log.*, 15(1):3:1–3:29, 2014. doi:10.1145/2490818.

12 Salem Derisavi, Holger Hermanns, and William H. Sanders. Optimal state-space lumping in markov chains. *Inf. Process. Lett.*, 87(6):309–315, 2003. doi:10.1016/S0020-0190(03)00343-0.

13 Ulrich Dorsch, Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Efficient coalgebraic partition refinement. In Roland Meyer and Uwe Nestmann, editors, *28th International Conference on Concurrency Theory, CONCUR 2017, September 5-8, 2017, Berlin, Germany*, volume 85 of *LIPIcs*, pages 32:1–32:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CONCUR.2017.32.

14 Agostino Dovier, Carla Piazza, and Alberto Policriti. An efficient algorithm for computing bisimulation equivalence. *Theor. Comput. Sci.*, 311(1-3):221–256, 2004. doi:10.1016/S0304-3975(03)00361-X.

15 Jan Friso Groote, Jan Martens, and Erik de Vink. Bisimulation by Partitioning Is $\Omega((m+n)\log n)$. In Serge Haddad and Daniele Varacca, editors, *32nd International Conference on Concurrency Theory (CONCUR 2021)*, volume 203 of *Leibniz International Proceedings*

in Informatics (LIPIcs), pages 31:1–31:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CONCUR.2021.31.

16 H. Peter Gumm. Functors for coalgebras. *Algebra Universalis*, 45(2):135–147, April 2001. doi:10.1007/s00012-001-8156-x.

17 H. Peter Gumm. On minimal coalgebras. *Applied Categorical Structures*, 16(3):313–332, June 2008. doi:10.1007/s10485-007-9116-1.

18 H. Peter Gumm and Tobias Schröder. Monoid-labeled transition systems. In *Coalgebraic Methods in Computer Science, CMCS 2001*, volume 44(1) of *ENTCS*, pages 185–204. Elsevier, 2001. doi:10.1016/S1571-0661(04)80908-3.

19 H. Peter Gumm and Tobias Schröder. Types and coalgebraic structure. *algebra universalis*, 53(2):229–252, 2005. doi:10.1007/s00012-005-1888-2.

20 Ichiro Hasuo, Bart Jacobs, and Ana Sokolova. Generic trace theory. In Neil Ghani and John Power, editors, *Proceedings of the Eighth Workshop on Coalgebraic Methods in Computer Science, CMCS 2006, Vienna, Austria, March 25-27, 2006*, volume 164 of *Electronic Notes in Theoretical Computer Science*, pages 47–65. Elsevier, 2006. doi:10.1016/j.entcs.2006.06.004.

21 John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

22 Thomas Ihringer. *Algemeine Algebra. Mit einem Anhang über Universelle Coalgebra von H. P. Gumm*, volume 10 of *Berliner Studienreihe zur Mathematik*. Heldermann Verlag, 2003.

23 Bartek Klin and Vladimiro Sassone. Structural operational semantics for stochastic and weighted transition systems. *Inf. Comput.*, 227:58–83, 2013. doi:10.1016/j.ic.2013.04.001.

24 Barbara König and Sebastian Küpper. Generic partition refinement algorithms for coalgebras and an instantiation to weighted automata. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2014. doi:10.1007/978-3-662-44602-7_24.

25 Alexander Kurz. *Logics for Coalgebras and Applications to Computer Science*. PhD thesis, Ludwig-Maximilians-Universität München, July 2000. URL: https://www.cs.le.ac.uk/people/akurz/LMU/Diss/all-s.ps.gz.

26 Alexander Kurz, Daniela Petrisan, Paula Severi, and Fer-Jan de Vries. Nominal coalgebraic data types with applications to lambda calculus. *Log. Methods Comput. Sci.*, 9(4), 2013. doi:10.2168/LMCS-9(4:20)2013.

27 Stefan Milius, Dirk Pattinson, and Thorsten Wißmann. A new foundation for finitary corecursion and iterative algebras. *Information and Computation*, page 104456, September 2019. doi:10.1016/j.ic.2019.104456.

28 Stefan Milius, Lutz Schröder, and Thorsten Wißmann. Regular behaviours with names - on rational fixpoints of endofunctors on nominal sets. *Appl. Categorical Struct.*, 24(5):663–701, 2016. doi:10.1007/s10485-016-9457-8.

29 Robert Paige and Robert Endre Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, 1987. doi:10.1137/0216062.

30 Jurriaan Rot. Coalgebraic minimization of automata by initiality and finality. In Lars Birkedal, editor, *The Thirty-second Conference on the Mathematical Foundations of Programming Semantics, MFPS 2016, Carnegie Mellon University, Pittsburgh, PA, USA, May 23-26, 2016*, volume 325 of *Electronic Notes in Theoretical Computer Science*, pages 253–276. Elsevier, 2016. doi:10.1016/j.entcs.2016.09.042.

31 Jan J. M. M. Rutten. Universal coalgebra: a theory of systems. *Theor. Comput. Sci.*, 249(1):3–80, 2000. doi:10.1016/S0304-3975(00)00056-6.

32 Věra Trnková. On a descriptive classification of set functors I. *Commentationes Mathematicae Universitatis Carolinae*, 12(1):143–174, 1971.

33 Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland,*

June 29 - July 2, 1997, pages 280–291. IEEE Computer Society, 1997. doi:10.1109/LICS.1997.614955.

34 Antti Valmari. Bisimilarity minimization in o(m logn) time. In Giuliana Franceschinis and Karsten Wolf, editors, *Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, June 22-26, 2009. Proceedings*, volume 5606 of *Lecture Notes in Computer Science*, pages 123–142. Springer, 2009. doi:10.1007/978-3-642-02424-5_9.

35 Antti Valmari and Giuliana Franceschinis. Simple $O(m \log n)$ time markov chain lumping. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 2010. doi:10.1007/978-3-642-12002-2_4.

36 Thorsten Wißmann, Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Efficient and modular coalgebraic partition refinement. *Log. Methods Comput. Sci.*, 16(1), 2020. doi:10.23638/LMCS-16(1:8)2020.

37 Thorsten Wißmann, Jérémy Dubut, Shin-ya Katsumata, and Ichiro Hasuo. Path category for free - open morphisms from coalgebras with non-deterministic branching. In Mikolaj Bojanczyk and Alex Simpson, editors, *Foundations of Software Science and Computation Structures - 22nd International Conference (FOSSACS 2019), Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11425 of *Lecture Notes in Computer Science*, pages 523–540. Springer, 2019. doi:10.1007/978-3-030-17127-8_30.

38 Thorsten Wißmann. *Coalgebraic Semantics and Minimization in Sets and Beyond*. Phd thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2020. URL: https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/14222.

39 Thorsten Wißmann, Stefan Milius, Shin-ya Katsumata, and Jérémy Dubut. A coalgebraic view on reachability. *Commentationes Mathematicae Universitatis Carolinae*, 60:4:605–638, December 2019. doi:10.14712/1213-7243.2019.026.

Appendix: Omitted Proofs

Proof of Corollary 2.6

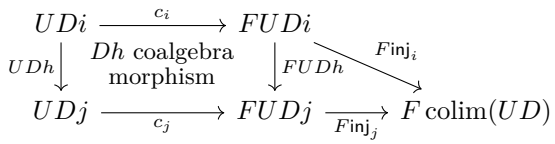
Let $c_i \colon UDi \to FUDi$ be the coalgebra structure of $Di \in \mathsf{Coalg}(F)$ for every $i \in \mathcal{D}$. For the colimit of $UD \colon \mathcal{D} \to \mathcal{C}$

$$UDi \xrightarrow{\mathsf{inj}_i} \operatorname{colim}(UD) \qquad \text{for every } i \in \mathcal{D}$$

apply F and precompose with c_i, yielding

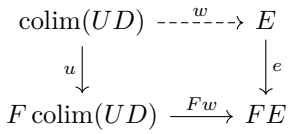
$$UDi \xrightarrow{c_i} FUDi \xrightarrow{F\mathsf{inj}_i} F\operatorname{colim}(UD) \qquad \text{for every } i \in \mathcal{D}.$$

This is a cocone for the diagram D because for all $h \colon i \to j$ in $\mathcal{D}$ the outside of the following diagram commutes:



Thus we obtain a coalgebra structure $u \colon \operatorname{colim}(UD) \to F\operatorname{colim}(UD)$. Since u is a cocone-morphism, every inj_i is an F-coalgebra morphism.

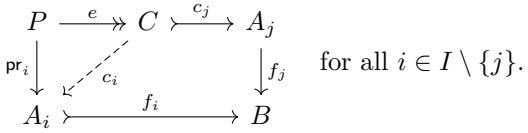
In order to show that $(\mathrm{colim}(UD), u)$ is the colimit of $D\colon \mathcal{D} \to \mathsf{Coalg}(F)$, consider another cocone $(m_i\colon Di \to (E,e))_{i\in\mathcal{D}}$.



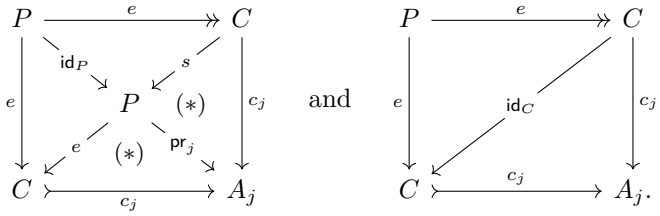
In $\mathcal{C}$, we obtain a cocone morphism $w\colon \mathrm{colim}(UD) \to E$. With a similar verification as before, $(e\cdot m_i\colon UDi \to FE)_{i\in\mathcal{D}}$ is a cocone for D, and thus both $d\cdot w$ and $Fw\cdot u\colon \mathrm{colim}(UD) \to FE$ are cocone morphisms (for UD). Since $\mathrm{colim}(UD)$ is the colimit, this implies that $d\cdot w = Fw\cdot u$, i.e. $w\colon (colim(UD), u) \to (E,e)$ is a coalgebra morphism. Since $U\colon \mathsf{Coalg}(F) \to \mathcal{C}$ is faithful, w is the unique cocone morphism, and so $(\mathrm{colim}\, UD, u)$ is indeed the colimit of UD. ◀

Proof of Lemma 2.16

Consider the $(\mathcal{E},\mathcal{M})$-factorization of pr_j into $e\colon P \twoheadrightarrow C$ and $m\colon C \rightarrowtail A_j$ with $\mathsf{pr}_j = m\cdot e$. On the image, we define a cone structure $(c_i\colon C \to A_i)_{i\in I}$ by $c_j = m$ and for every $i \in I\setminus\{j\}$ by the diagonal fill-in:



The diagonal c_i is induced, because $f_i \in \mathcal{M}$ for all $i \in I\setminus\{j\}$. The family $(c_i)_{i\in I}$ forms a cone for the wide pullback, because for all $i,i' \in I$ we have $f_i\cdot c_i = f_j\cdot c_j = f_{i'}\cdot c_{i'}$. This makes e a cone morphism, because $c_i\cdot e = \mathsf{pr}_i$ for all $i\in I$. Moreover, the limiting cone P induces a cone morphism $s\colon C \to P$ and we have $s\cdot e = \mathsf{id}_P$. Consider the commutative diagrams:

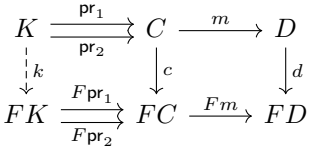


The parts marked by $(*)$ commute because e and s are cone morphisms. Since the diagonal fill-in in Definition 2.10.3 is unique, we have $e\cdot s = \mathsf{id}_C$. Thus, e is an isomorphism, and $\mathsf{pr}_j = c_j\cdot e$ is in $\mathcal{M}$, as desired. ◀

Proof of Lemma 3.2

It is clear that every Mono-carried homomorphism is monic in $\mathsf{Coalg}(F)$. Conversely, let $m\colon (C,c) \to (D,d)$ be a monomorphism in $\mathsf{Coalg}(F)$. Let $\mathsf{pr}_1, \mathsf{pr}_2\colon K \to C$ be a weak kernel pair of m. Since F preserves weak kernel pairs, $F\mathsf{pr}_1, F\mathsf{pr}_2\colon FK \to FC$ is a weak kernel pair of $Fm\colon FC \to FD$. This induces some cone morphism $k\colon K \to FK$ making pr_1 and pr_2

coalgebra morphisms $(K,k) \to (C,c)$:

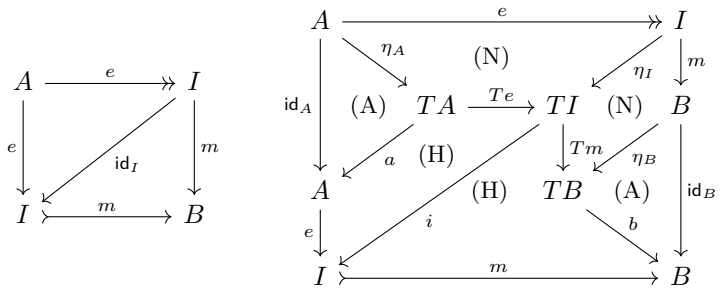


Since m is monic in $\mathsf{Coalg}(F)$, this implies that $\mathsf{pr}_1 = \mathsf{pr}_2$. For the verification that m is a monomorphism in $\mathcal{C}$, consider $f, g\colon X \to C$ with $m \cdot f = m \cdot g$. Since $\mathsf{pr}_1, \mathsf{pr}_2$ is a weak kernel pair, it induces some cone morphism $v\colon X \to K$, fulfilling $f = \mathsf{pr}_1 \cdot v$ and $g = \mathsf{pr}_2 \cdot v$. Since, $\mathsf{pr}_1 = \mathsf{pr}_2$, we find $f = g$ as desired. ◀

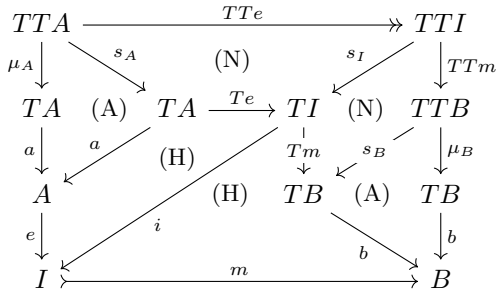
Proof of Lemma 3.6

Denote the unit and multiplication of the monad T by $\eta\colon \mathsf{Id} \to T$ and $\mu\colon TT \to T$, respectively. Consider an T-algebra homomorphism $f\colon (A,a) \to (B,b)$ for Eilenberg-Moore algebras (A,a) and (B,b) and denote its image factorization in $\mathsf{Alg}(T)$ by (I,i), with homomorphisms $e\colon (A,a) \twoheadrightarrow (I,i)$ and $m\colon (I,i) \rightarrowtail (B,b)$. We verify that $i\colon TI \to I$ is an Eilenberg-Moore algebra.

- First, we verify $i \cdot \eta_I = \mathsf{id}_I$ by showing that both $i \cdot \eta_I$ and id_I are both diagonals of the following square:



  The left-hand square commutes trivially, and the right-hand square commutes because η is natural (N), because e and m are T-algebra homomorphisms (H), and because (A,a) and (B,b) are Eilenberg-Moore algebras (A). By the uniqueness of the diagonal lifting property, we obtain $i \cdot \eta_I = \mathsf{id}_I$.
- Next, we verify $i \cdot Ti = i \cdot \mu_i$ by showing that both are diagonals in same square. To this end, let $(s_A, s_I, s_B) \in \{(Ta, Ti, Tb), (\mu_A, \mu_I, \mu_B)\}$, i.e. we obtain one diagonal for (Ta, Ti, Tb) and one for (μ_A, μ_I, μ_B):



  In both cases, we have that e and m are T-algebra homomorphisms (H). For $(s_A, s_I, s_B) := (Ta, Ti, Tb)$, we have that (A,a) and (B,b) are Eilenberg-Moore algebras (A) and that e

and m are homomorphisms (N). For $(s_A, s_I, s_B) := (\mu_A, \mu_I, \mu_B)$, we have that the parts (A) commute trivially, and the parts (N) commute because μ is natural. Since T preserves $\mathcal{E}$, we have $TTe \in \mathcal{E}$, and thus by the uniqueness of the diagonal, we obtain $i \cdot Ti = i \cdot \mu_I$. Thus, (I, i) fulfils the axioms of an Eilenberg-Moore algebra. The remaining properties of the factorization system hold because the Eilenberg-Moore category is a full subcategory of $\mathsf{Alg}(T)$. ◀

Proof of Lemma 4.8

We verify the postulated equivalence using that C is $\mathcal{M}$-minimal iff every $h\colon B \to C$ is an epimorphism (Lemma 4.6, $\mathcal{E} = \mathsf{Epi}$).

- For 'if', we verify that every $h\colon B \to C$ is an epimorphism: for $u, v\colon C \to D$ with $u \cdot h = v \cdot h$, we directly obtain $u = v$ by assumption. Thus, h is an epimorphism.
- For 'only if', consider $u, v\colon C \to D$ and take a weak equalizer $e\colon E \to C$; hence, $u \cdot e = v \cdot e$. Since e is an epimorphism (by minimality), we obtain $u = v$. ◀