The Provenance of Elegance in Computation – Essays Dedicated to Val Tannen

Tannen's Festschrift, May 24–25, 2024,
University of Pennsylvania, Philadelphia, PA, USA

Edited by

Antoine Amarilli
Alin Deutsch

OASICS

Editors

Antoine Amarilli
Télécom Paris, France
a3nm@a3nm.net

Alin Deutsch
University of California, San Diego, USA
abdeutsch@ucsd.edu

ACM Classification 2012
Information systems

ISBN 978-3-95977-320-1

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/978-3-95977-320-1.

Publication date
June, 2024

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/OASIcs.Tannen.2024.0

OASIcs – OpenAccess Series in Informatics

ISSN 1868-8969

https://www.dagstuhl.de/oasics

To Val Tannen

Contents

Preface

This Festschrift volume accompanies a colloquium held at the University of Pennsylvania on May 24–25, 2024 in celebration of the distinguished career of Val Tannen. Attendants gathered from all over the world to express their admiration for Val as a researcher as well as their love for him as a person. We are lucky to have him as a role model: a teacher, a mentor, a collaborator, a colleague, a friend (several categories often apply).

The articles presented here are scientific offerings from some of us to Val. They pertain to some of the many areas of his research interests. Our one regret is that, to preserve the element of surprise, we could not collaborate with Val on these papers, nor could we at least ask for his feedback. This is a pity, as Val is famously unfailingly insightful, always honest, and extremely generous with his time.

Val has contributed seminally to the principles of both programming languages and databases and also to the cross-pollination and unification of the two areas. He also contributed to bioinformatics and to systematic and evolutionary biology. For lack of space, we cannot do justice here to his manifold contributions, and can only include a few highlights.

One of Val's major contributions is the use of *structural recursion*, together with other ideas from functional programming and type theory, to inform the design of query languages for post-relational data. Besides providing the theoretical underpinning for query optimization over nested-relational, complex-valued and object-oriented data, his work yielded – through the use of comprehensions – a standard technique for embedding relational databases in programming languages. Modern database systems support user-defined aggregates using a template that is an instance of Val's techniques.

Val was instrumental in unifying a series of classic database optimization techniques that had been previously developed independently and implemented in different phases of the optimizer, with only limited interaction. Examples include rewriting using materialized views, join minimization, semantic optimization, index- and hash-based query evaluation, all of which were unified by reduction to query minimization under constraints. This enabled a novel, chase-based optimization approach in which these techniques, as well as other techniques that had not been explicitly articulated, are implicitly considered simultaneously. This allows them to feed off each other synergistically to yield plans that standard phase-based optimization will necessarily miss even if given unbounded computational resources. Val's work on chase-based optimization brought a purely abstract concept, the "chase", introduced for theoretical studies of logical constraints, to the attention of developers of query optimizers.

A highly celebrated outcome of Val's work is the invention of *provenance semirings*, which yield a generalization of many adjuncts to relational databases, such as probabilistic databases, C-tables, bag semantics, and even database security, enabling their unified treatment. In addition, provenance semirings provide a widely adopted general formalism for defining, capturing, storing, reasoning about, and optimizing data provenance. By now, the elegant concept of K-relations is widely known, well beyond the database research community, and has been applied in domains as diverse as operating systems, programming languages, and verification. Closer to home, Val's work has inspired database researchers to use K-relations as a tool in analyzing the fine-grained complexity of query evaluation, or to extend relational query optimization techniques to tensor processing systems.

Val's work features a common leitmotif: the surprising unification of seemingly disparate concepts and theories. Such unification is not achieved by devising complicated hybrid unions of these theories, but rather by distilling them down to their essence in sublimely elegant style, thus exposing their commonality. Unsurprisingly, some of his PhD students nicknamed Val the Great Unifier. We hereby dub him the Amazingly Insightful Great Unifier, a well-deserved title whose acronym is moreover a nod to the sharp wit that earned him a reputation as a delightful conversationalist. We are looking forward to many more conversations with him, both scientific and social in nature.

May 2024

Antoine Amarilli
Peter Buneman
Daniel Deutch
Alin Deutsch
Zack Ives
Dan Suciu

Also on behalf of the contributing authors listed below.

List of Authors

Paolo Atzeni (1)
Università Roma Tre, Italy

Teodoro Baldazzi (1)
Università Roma Tre, Italy

Luigi Bellomarini (1)
Banca d'Italia, Roma, Italy

Siddharth Bhaskar (2)
Department of Computer Science, James Madison University, Harrisonburg, VA, USA

Sophie Brinke (3)
RWTH Aachen University, Germany

Peter Buneman (4)
University of Edinburgh, UK

Stefano Ceri (1)
Politecnico di Milano, Italy

Vassilis Christophides (8)
ETIS Laboratory, CY Cergy Paris Université, ENSEA, France

Andrea Colombo (1)
Politecnico di Milano, Italy

Alin Deutsch (5)
University of California, San Diego, CA, USA

Wenfei Fan (6)
Shenzhen Institute of Computing Sciences, China; University of Edinburgh, UK;
Beihang University, Beijing, China

Jean Gallier (7)
University of Pennsylvania, Philadelphia, PA, USA

Andrea Gentili (1)
Banca d'Italia, Roma, Italy

Erich Grädel (3)
RWTH Aachen University, Germany

Steven Lindell (2)
Department of Computer Science, Haverford College, PA, USA

Shuhao Liu (6)
Shenzhen Institute of Computing Sciences, China

Lovro Mrkonjić (3)
RWTH Aachen University, Germany

Nikolaos Myrtakis (8)
Department of Computer Science, University of Crete, Heraklion, Greece;
ETIS Laboratory, CY Cergy Paris Université, ENSEA, France

Matthias Naaf (3)
RWTH Aachen University, Germany

Emanuel Sallinger (1)
TU Wien, Austria; University of Oxford, UK

Pierre Senellart (9)
DI ENS, ENS, PSL University, CNRS, Paris, France;
Inria, Paris, France;
Institut Universitaire de France, Paris, France;
CNRS@CREATE LTD, Singapore;
IPAL, CNRS, Singapore

Dan Suciu (10)
University of Washington, Seattle, WA, USA

Ioannis Tsamardinos (8)
Department of Computer Science, University of Crete, Heraklion, Greece

Stijn Vansummeren (4)
UHasselt, Data Science Institute, Belgium

Scott Weinstein (2)
Department of Philosophy, University of Pennsylvania, Philadelphia, PA, USA

Limsoon Wong (11)
School of Computing, National University of Singapore, Singapore

Explaining Enterprise Knowledge Graphs with Large Language Models and Ontological Reasoning

Teodoro Baldazzi
Università Roma Tre, Italy

Luigi Bellomarini
Banca d'Italia, Roma, Italy

Stefano Ceri
Politecnico di Milano, Italy

Andrea Colombo
Politecnico di Milano, Italy

Andrea Gentili
Banca d'Italia, Roma, Italy

Emanuel Sallinger
TU Wien, Austria
University of Oxford, UK

Paolo Atzeni
Università Roma Tre, Italy

Abstract

In recent times, the demand for transparency and accountability in AI-driven decisions has intensified, particularly in high-stakes domains like finance and bio-medicine. This focus on the provenance of AI-generated conclusions underscores the need for decision-making processes that are not only transparent but also readily interpretable by humans, to built trust of both users and stakeholders. In this context, the integration of state-of-the-art Large Language Models (LLMs) with logic-oriented Enterprise Knowledge Graphs (EKGs) and the broader scope of Knowledge Representation and Reasoning (KRR) methodologies is currently at the cutting edge of industrial and academic research across numerous data-intensive areas. Indeed, such a synergy is paramount as LLMs bring a layer of adaptability and human-centric understanding that complements the structured insights of EKGs. Conversely, the central role of ontological reasoning is to capture the domain knowledge, accurately handling complex tasks over a given realm of interest, and to infuse the process with transparency and a clear provenance-based explanation of the conclusions drawn, addressing the fundamental challenge of LLMs' inherent opacity and fostering trust and accountability in AI applications. In this paper, we propose a novel neuro-symbolic framework that leverages the underpinnings of provenance in ontological reasoning to enhance state-of-the-art LLMs with domain awareness and explainability, enabling them to act as natural language interfaces to EKGs.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning; Computing methodologies → Natural language processing; Theory of computation → Data provenance

Keywords and phrases provenance, ontological reasoning, language models, knowledge graphs

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.1

Funding The work on this paper was partially supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201, 10.47379/VRG18013, 10.47379/NXT22018]; and the Christian Doppler Research Association (CDG) JRC LIVE.
Stefano Ceri: S.C. is supported by the PNRR-PE-AI FAIR project, funded by the NextGenerationEU program.
Andrea Colombo: A.C. kindly acknowledges INPS for the funding of his Ph.D. program.

1 Introduction

In today's data-driven industrial landscape, adhering to the principles of Fairness, Accountability, Transparency, and Ethics (FATE) has become paramount for AI applications [44]. Indeed, the absence of transparency in AI's decision-making processes prevents stakeholders and users from assessing their fairness, detecting potential biases, and verifying their overall reliability. This is particularly relevant in high-stakes domains such as finance and biomedicine, where there is an increasing demand for a clear, comprehensible, natural language explanation of AI's conclusions (i.e., their *provenance*) to back trustworthy critical decisions. Such a feature would act as a bridge between the opaque inner workings of AI and human comprehension, fostering informed decision-making and mitigating risks associated with black-box models, in favour of users' trust and adherence to ethical standards [35].

This requirement has gained further traction with the recent breakthrough of AI-based chatbots and Large Language Models (LLMs) [46], which has marked a significant turning point in the field of Natural Language Processing (NLP) and a pivotal shift in the access to data and knowledge towards more natural, user-friendly, and high-level paradigms. Notably, LLMs such as OpenAI's GPT [55] and Meta's Llama [61] have transcended traditional academic and industrial applications, capturing the general public's attention towards generative AI capabilities. However, concerns persist regarding their lack of factual knowledge and accuracy over enterprise domains, even when fine-tuning is involved [5], and, more importantly, their opaqueness due to a very limited explainability of their conclusions [66].

Conversely, traditional *Knowledge Representation and Reasoning* (KRR) [48] approaches are inherently *domain-aware* and explainable [24]. Indeed, logic-based reasoning in query answering, often referred to as *ontological reasoning* [22], allows for FATEness, as it is designed to provide factual conclusions augmented with the logically consequential steps, in the form of top-down logical inference, that led to such results [25, 24]. For this reason, in the database and the AI communities, we are observing the surge of increasingly mature, efficient, and scalable *intelligent systems with reasoning capabilities*, backed by expressive logic-based KRR formalisms. Among them, database query languages based on logic programming, such as Datalog and its extensions [1, 22, 20, 21], are a yardstick for AI systems rooted in ontological reasoning, thanks to their effective trade-off between expressive power and computational complexity [11, 28, 48]. Leveraging such systems, domain-specific knowledge can be captured by combining factual data from corporate databases with business-level definitions as ontologies in *Enterprise Knowledge Graphs* (EKGs), and further augmented by reasoning over them. Despite this, the interaction remains query-based, often operating at a low level and lacking flexibility, thus proving itself challenging for non-specialists.

In light of these considerations, it becomes clear why *neuro-symbolic* methodologies are at the forefront of academic and industrial interest. Indeed, their goal to synergistically combine the intrinsic domain-expertise and transparency of deductive systems with the power of LLMs in understanding and generating fluent and interpretable text holds immense potential to build more intelligent, versatile, and explainable AI-based applications on KGs, paving the way for a new era of transparent data-driven decision-making within organizations [54, 31, 42].

With the goal of contributing to such a pivotal challenge, this paper strives to strengthen LLMs in their use as explainable NL interfaces to EKGs by leveraging the power of ontological reasoning. In simple terms, our goal consists in *enabling the answer to "why this conclusion" questions in natural language and posed by the user over an EKG.* We operate in the context of the VADALOG [14, 8] system, a Datalog-based reasoning engine for knowledge graphs, that finds many industrial applications [12, 15, 10, 4, 3, 32, 62]. We employ VADALOG to explore the factual information derived by applying the domain rules in high-stakes domains of interest, via the well-known CHASE procedure [53] of databases. This enables us to augment LLAMA 2-70B models with the derived domain knowledge and the provenance of the inferred conclusions, while preserving their human-like orientation and flexibility at handling question-answering tasks in natural language.

More in detail, our **contributions** can be summarized as follows.

- We present a **chase verbalization technique** to enhance the domain expertise and explanation capabilities of LLMs by leveraging ontological reasoning over knowledge graphs and use cases in relevant domains.
- We deliver such an approach in KGLM, a **novel neuro-symbolic pipeline** to build LLM-powered explainable interfaces to EKGs by interacting with CHASE-based reasoning engines such as the VADALOG system.
- We provide a **practical application** of KGLM by integrating it within KG-ROAR [13, 6], a framework we developed to showcase VADALOG reasoning on financial use cases.

Related Work. Numerous studies have been conducted to investigate different aspects of provenance, namely its tracking, storing, and presentation [18, 26, 36, 58, 43, 47]. A longstanding challenge is dealing with the complexity of provenance expressions, with the goal of presenting them in a user-comprehensible form. To this aim, semirings models [40, 39, 56, 59] are the benchmark for their ability to present the provenance in an efficient and mathematically elegant form. These structures capture the two key aspects of data usage in provenance: joint contribution (represented by addition) and alternative sources (represented by multiplication). This allows for a concise representation of how various inputs contribute to the final output, enabling reasoning about aspects like confidence, access control, and cost associated with the data lineage. However, this kind of representation is still not easily accessible by non-expert users. In other studies, some graph-based representations of the provenance have been proposed [23, 27, 60], allowing, for instance, user control over the provenance graph, i.e., by visually tracking contributors and sources. While they have the advantage of enhancing user interaction and inspection, they are still not the most natural way of interacting and understanding the provenance. A more recent line of research attempts to present provenance and answer to queries in natural language [33, 29, 49, 30, 19]. In most cases, these efforts only support queries of low complexity and the NL sentence depends on the quality of an input query asked in natural language. In other cases, explanations are fragmented, presenting rules without a cohesive narrative [19]. Additionally, linking provenance representation to an input query does not allow unlocking innovative uses of such precious information, such as exploiting it for generating a training corpus for LLMs. Current models are very effective for general tasks, but often struggle when it comes to specific domains. For example, it has been shown that FinBert [51] and BloombergGPT [64], two fine-tuned models on financial textual data, outperform generic LLMs on question-answering tasks. In such a context, provenance information from business reasoning tasks could be a valuable resource for generating a corpus of domain-specific knowledge to be injected into an LLM for fine-tuning or via *Retrieval-Augmented Generation* [50] (RAG) mechanisms.

Overview. The remainder of this paper is organized as follows. In Section 2 we provide essential background notions on ontological reasoning and introduce the VADALOG system. In Section 3 we illustrate the verbalization technique for LLM explanation and the KGLM pipeline. Section 4 delves into the application of KGLM within KG-ROAR. Our conclusions are drawn in Section 5.

2 Ontological Reasoning in the Vadalog System

To guide our discussion, we first lay out some preliminary notions on ontological reasoning over enterprise knowledge graphs, with a specific focus on the Vadalog system and the chase procedure at its foundation.

Relational Foundations. Let $\mathbf{C}$ and $\mathbf{V}$ be disjoint countably infinite sets of *constants* and *variables*, respectively. A (*relational*) *schema* $\mathbf{S}$ is a finite set of relation symbols (or *predicates*) with associated arity. A *term* is either a constant or a variable. An atom over $\mathbf{S}$ is an expression of the form $R(\bar{v})$, where $R \in \mathbf{S}$ is of arity $n > 0$ and $\bar{v}$ is an n-tuple of terms. A *database* (*instance*) over $\mathbf{S}$ associates to each symbol in $\mathbf{S}$ a relation of the respective arity over the domain of constants. The members of the relations are called *tuples* or *facts*.

Dependencies. A Vadalog program consists of a set of tuples and *tuple-generating dependencies* (TGDs), i.e., function-free Horn clauses of the form $\forall\bar{x}\forall\bar{y}(\varphi(\bar{x},\bar{y})\rightarrow\exists\bar{z}\ \psi(\bar{x},\bar{z}))$, where $\varphi(\bar{x},\bar{y})$ (the *body*) and $\psi(\bar{x},\bar{z})$ (the *head*) are conjunctions of atoms over the respective predicates, $\bar{x},\bar{y}$ are vectors of universally quantified variables and constants, and $\bar{z}$ is a vector of existentially quantified variables. Quantifiers can be omitted and conjunction is denoted by comma. A predicate is *intensional* (IDB) if it occurs in at least one head of the schema $\mathbf{S}$, otherwise it is *extensional* (EDB) [1, 25, 38]. A fact corresponding to an intensional predicate is intensional, otherwise it is extensional.

Vadalog Extensions. Real-world applications may require support for multiple features that extend the declarative language. Among them, aggregate functions, namely *sum*, *prod*, *min*, *max* and *count*, as well as SQL-like grouping constructs, are particularly relevant. In the Vadalog context, support for aggregate functions is achieved by means of *monotonic aggregations* [63]. Other essential extensions, integrated in Vadalog to address real-world scenarios, include *negations* and negative constraints of the form $\varphi(\bar{x},\bar{y})\rightarrow\perp$, where $\varphi(\bar{x},\bar{y})$ is a conjunction of atoms and $\perp$ denotes the truth constant false to model disjointness or non-membership, as well as *expressions* in rule bodies, modelled with comparison $(>,<,\geq,\leq,\neq)$ and algebraic $(+,-,*,/,$ etc.$)$ operators.

Reasoning Task. KRR approaches model KGs as the combination of an *extensional component*, essentially the ground business data in a database, and an *intensional component*, which formally describes the business knowledge as a set of rules in a declarative language such as Vadalog. Performing ontological reasoning over the KG augments it with new inferred knowledge derived from the application of the rules over the business data. More formally, given a database D and the query $Q = (\Sigma, Ans)$, where Σ is the set of rules and Ans an n-ary predicate, a reasoning task consists of finding an instance J such that a tuple $\bar{t} \in J$ if and only if $\bar{t} \in Q(D)$ and for every other instance J' such that $\bar{t} \in J'$ if and only if $\bar{t} \in Q(D)$, there is a homomorphism h from J to J'.

Chase Procedure. The semantics of a Vadalog program can be defined in an operational way with the *chase procedure* [45, 53]. It enforces the satisfaction of a set Σ of rules over a database D, incrementally augmenting D with facts entailed via the application of the rules over D, until fixpoint. While Vadalog guarantees that such fixpoint exists when only the core features are used [14], the joint presence of algebraic operations and recursion must be carefully handled, as even simple Datalog programs can be in general non-terminating [1].

A TGD $\sigma : \varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{z})$ is applicable to D if there exists a homomorphism θ such that $\theta(\varphi(\bar{x}, \bar{y})) \subseteq D$. Then, a *chase step* adds the fact $\theta(\psi(\bar{x}, \bar{z}))$ to D, if not already in D. The *chase graph* $\mathcal{G}(D, \Sigma)$ is the directed acyclic graph with the facts from the chase $\Sigma(D)$ as nodes and an edge from a node n to a node m if m derives from n (and possibly other facts) via a CHASE step [20]. A comprehensive examination of reasoning termination in VADALOG has been thoroughly explored in dedicated works [16, 9].

▶ **Example 1** (Trading Activity). Let us consider a simple trading activity managed with a *smart contract*. Here, D contains a log over time of buy/sell orders from the traders who invest in it as well as market information, e.g., asset prices (*Price*), or market shutdowns (*MarketClosed*). The following set Σ contains the VADALOG rules governing the basic functioning of the market, i.e., under which conditions the orders are accepted and how profits and losses are computed.

$$Open(x, y, t_1), \neg\, MarketClosed(t_1) \rightarrow Accepted(x, y, t_1) \qquad (\sigma_1)$$

$$Accepted(x, y, t_1), Price(p_1, t_1), k = y * p_1 \rightarrow Position(x, y, k, t_1) \qquad (\sigma_2)$$

$$\begin{aligned} &Close(x, t_2), Price(p_2, t_2), Position(x, y, k, t_1),\\ &\qquad t_2 > t_1, pl = y * p_2 - k \rightarrow Return(x,pl) \end{aligned} \qquad (\sigma_3)$$

If a trader x wants to open a position (buy) on a certain asset of size y at time t_1 and the market is open at t_1, the order is accepted (rule σ_1). If the order by x is accepted and the asset price at t_1 is p_1, then x holds a position on the market at time t_1 of size y and of notional (total value) k equal to $y * p_1$ (rule σ_2). If, later at t_2, trader x decides to close its position (sell) and the price at t_2 is p_2, then x gets returns (profits or losses) from its trading activity as $y * p_2 - k$ (rule σ_3).

Let us also consider an excerpt of database $D = \{Open(EGTech,0.3,1), Price(124,1), Price(147,9), Close(EGTech,9), MarketClosed(5)\}$. Figure 1 illustrates the CHASE graph derived from the activation of Σ over D. Specifically, rule σ_1 generates the fact *Accepted*(*EGTech*,0.3,1), as the market is not closed at time 1. Then, the fact *Position*(*EGTech*,0.3, 37.2,1) is derived via rule σ_2. Finally, as trader *EGTech* closes the position, i.e., sells the asset, at time 9 and the price goes up to 147\$, then *EGTech* gets a profit of 6.9\$ as return via rule σ_3.

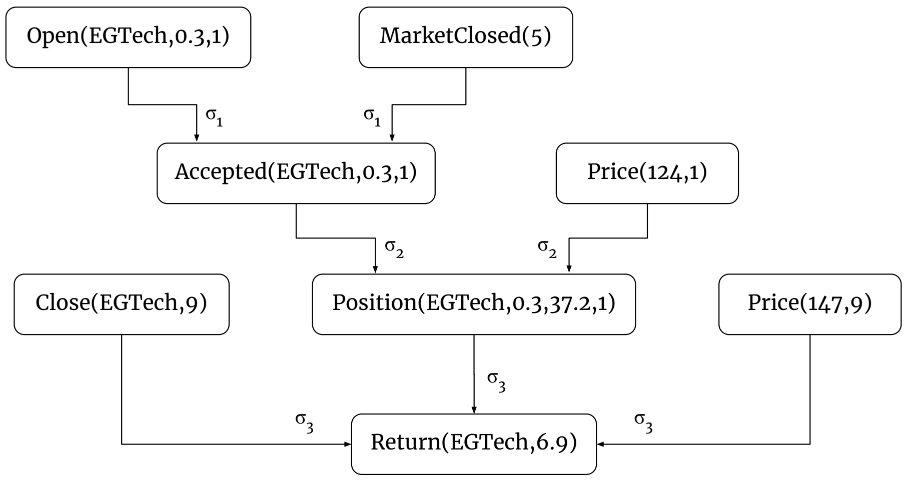


Figure 1 Instance of CHASE graph for Example 1.

The Vadalog System. The VADALOG system is a state-of-the-art ontological reasoning engine that leverages the theoretical underpinnings of the CHASE procedure and the vast experience of the database community on provenance to power efficient, scalable, and explainable reasoning tasks over critical business domains and large enterprise KGs [8].

To achieve this, it adopts a *streaming* data processing architecture based on the *pipes and filters* style [14, 7]. Here, the set of logic rules Σ and the queries are translated into active *data scans* (linear scans for linear TGDs, join scans for join TGDs, and an output scan for the query), connected by intermediate buffers in a processing pipeline. The reasoning process is performed as a data stream along the pipeline, where each filter (i.e., scan) reads tuples from the respective parent, from the output scan down to the external data stores that inject ground facts into the pipeline. Interactions between scans occur by means of primitives *open()*, *next()*, *get()*, *close()*, which open the parent stream, ask for the presence of a fact to fetch, obtain it, and close the communication, respectively. Since, for each filter, multiple parent filters may be available, VADALOG selects which one to invoke (via *next()* call) by employing specific *routing strategies* (*round-robin*, *shortest path*, etc.) that manage a priority queue of the sources. This methodology allows VADALOG to keep track of the provenance of each result, derived from one or more CHASE steps. Unlike traditional *semi-naive* approaches [1], VADALOG generalizes the *volcano iterator model* [37], operating in a *pull-based query-driven* fashion in which, ideally, facts are materialized only at the end of the CHASE procedure and if they contributed to the reasoning task.

Figure 2 illustrates VADALOG processing pipeline for the scenario in Example 1 given, as ontological reasoning task, the query q of finding the returns for the trader *EGTech*. Here the output filter o sends a *next()* message to the query processor, which propagates it to the TGDs scans. Note that each filter in the figure is labelled with the corresponding rule number from the above scenario.

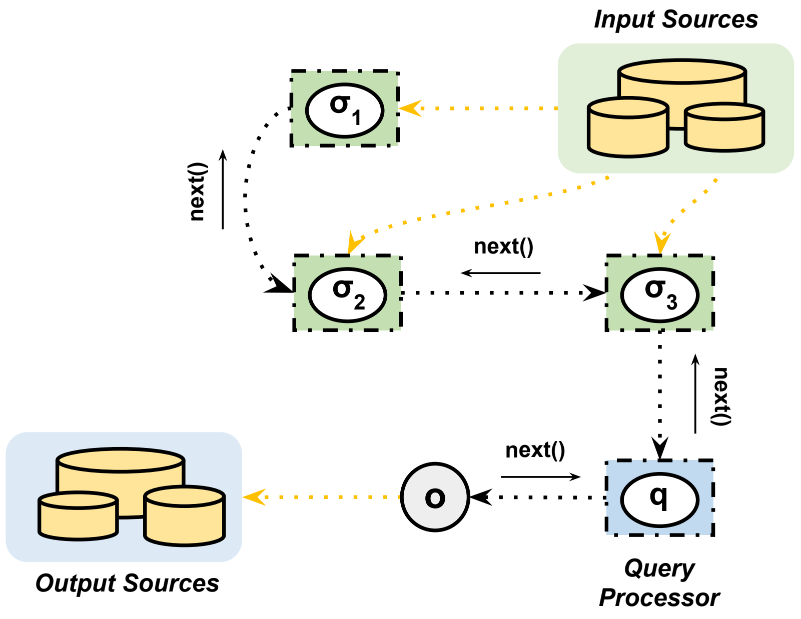


Figure 2 Processing pipeline of VADALOG for Example 1.

3 Overview of the KGLM Pipeline

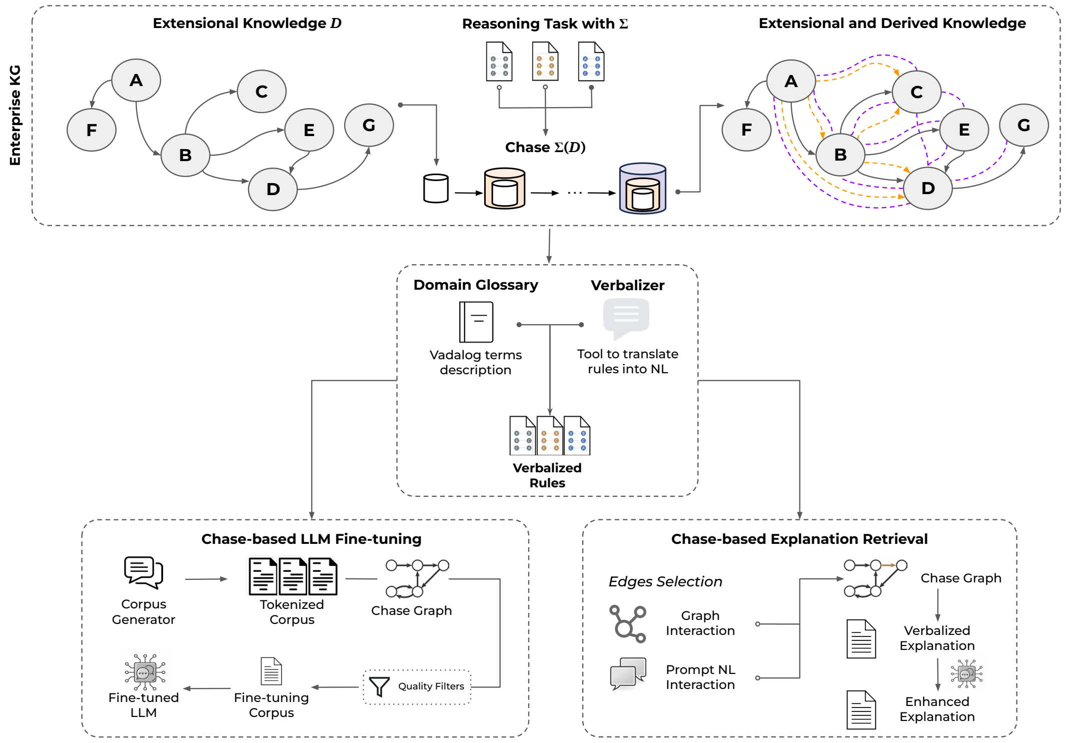


Figure 3 Visual overview of the KGLM pipeline.

Logic-oriented Enterprise Knowledge Graphs and Knowledge Representation and Reasoning approaches are at the forefront of explainable AI, which is as of today an area of paramount importance for both the academic community and the industry sector. Current research in KRR focuses on developing ontological reasoning systems, such as VADALOG, that effectively combine factual knowledge (extensional knowledge) with formally defined domain expertise (intensional knowledge) expressed through logical rules. In this context, integrating state-of-the-art Large Language Models with logic-oriented EKGs holds immense promise for developing explainable and human-oriented AI tools. A significant challenge, however, lies in the mismatch between the technologies. In fact, KRR systems require queries to be formulated in their specific formalisms, with knowledge generation restricted to what reasoning rules can capture. In contrast, LLMs lack the comprehensive domain models that are a cornerstone of KRR approaches. One promising approach involves leveraging the strengths of both paradigms in a synergistic fashion. Indeed, LLMs excel at natural language generation, allowing them to translate the complex relationships and entities within EKGs into human-understandable explanations. EKGs, in turn, provide LLMs with a foundation of factual knowledge, provenance, and ontological reasoning, thus mitigating the risk of inconsistencies and biases often present in LLMs' outputs.

Vadalog-powered KGLM. Specifically, we identify two main synergies between LLMs and logic-based EKGs. On the one hand, we can leverage the reasoning output, enriched by the provenance of each conclusion, to inject domain-specific knowledge for fine-tuning an LLM.

On the other hand, we can exploit the text manipulation capabilities of such models to enable a natural language navigation of the EKG, presenting its inferred edges and provenance in a human-like form. In the context of the VADALOG system, we recall that the provenance is provided by leveraging the CHASE procedure. Indeed, the CHASE graph resulting from a reasoning process corresponds to a directed acyclic graph representation of how each output fact was derived from the input facts by applying one or more VADALOG rules.

We develop such synergies between VADALOG-powered EKGs and LLMs in KGLM, the first, to the best of our knowledge, neuro-symbolic framework to enhance state-of-the-art LLMs with domain awareness and explainability, making them suitable to act as natural language interfaces to EKGs. Indeed, as further showcased later in this section and in Section 4, the features provided by KGLM are highly complementary, and have as their foundation a so-called *verbalizer* module that deterministically translates the KG into natural language sentences. Such *verbalizations* can then either be used as input to build question-answering fine-tuning corpora for LLMs, or they can be composed to retrieve the NL explanation of how a certain result was derived in the CHASE, which can then be injected into the LLM to achieve more fluent explanations that, in industrial applications, represent readable business reports. Indeed, these integrations can potentially enable a proper neuro-symbolic reasoning behavior by combining unstructured information with internal business rules to derive novel knowledge.

Reasoning Use Cases. To support our discussion, let us consider two scenarios involving a financial Enterprise Knowledge Graph. Specifically, we refer to an *ownership* KG, where the extensional component consists of *Owns* relationships, as depicted in Figure 4.

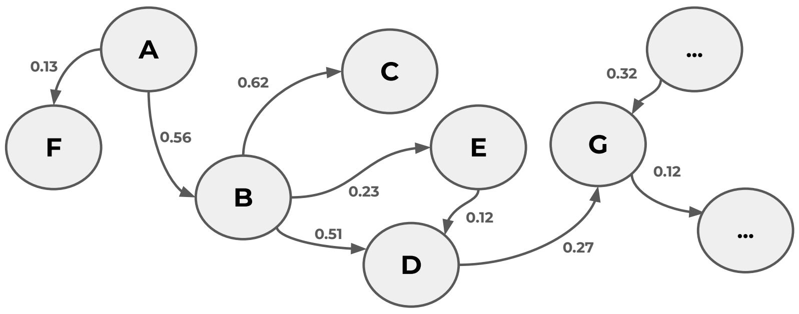


Figure 4 Portion of ownership knowledge graph. Nodes are companies. Solid edges are *owns* relationships with their shares.

On top of the KG, we can formalize an intensional component into VADALOG rules, which augment the graph with new knowledge in the form of novel edges.

▶ **Example 2** (Company Control). This scenario allows analysts to understand who has decision power in companies, based on who controls the majority of votes, in a "one-share one-vote" assumption. To this end, the task augments the ownership graph with "*control*" edges, as follows [41]: *A company x directly owning s shares of a company y, controls such shares via y itself* (rule σ_4). *If x controls a company z and z owns s shares of y, then x controls s shares of y via z* (rule σ_5). *Finally, if x controls the majority of the shares of y, directly or indirectly, then x controls y* (rule σ_6).

$$Owns(x, y, s) \rightarrow ControlledShares(x, y, y, s) \qquad (\sigma_4)$$

$$Control(x, z), Owns(z, y, s) \rightarrow ControlledShares(x, z, y, s) \qquad (\sigma_5)$$

$$ControlledShares(x, \_, y, s), ts = sum(s), ts > 0.5 \rightarrow Control(x, y) \qquad (\sigma_6)$$

▶ **Example 3** (Close Link). Integrated ownership refers to the total stake a single entity holds in another entity, considering both direct and indirect ownership throughout the graph, with finite or infinite paths [52]. The value of integrated ownership I is calculated as $lim_{\epsilon \to 0} \sum_{P_i \in B_\epsilon} w_\epsilon(P_i)$, where B_ϵ is the set of all paths $P = [x, p_1, \ldots, p_k, y]$ in the ownership graph such that $x \neq p_i$ for $i = 1, \ldots, k$, and where $w_\epsilon(P) = \Pi_{(p_i, p_j) \in P} w(p_i, p_j) > \epsilon$, with $w(p_i, p_j)$ representing the direct ownership of p_i on p_j, $\epsilon \in \mathbb{R}^+$, and $0 < \epsilon \leq 1$. Note that integrated ownership differs from simple ownership used in Example 2. Applying this formulation to the regulation of the European Central Bank [34], we can say that x is in *close link* with y if: *(i) the integrated ownership of x on y is at least* 20% (rule σ_7); *(ii) y is in close link with x* (rule σ_8); *(iii) there is a third company z, whose integrated ownership on x and y is at least* 20% (rule σ_9).

$$IntOwns(x, y, s), s > 0.2 \to CloseLink(x, y) \tag{σ_7}$$

$$CloseLink(x, y) \to CloseLink(y, x) \tag{σ_8}$$

$$\begin{aligned} &IntOwns(x, y, s_1), IntOwns(x, z, s_2), s_1 > 0.2, s_2 > 0.2, \\ &\quad x \neq y, x \neq z, y \neq z \to CloseLink(y, z) \end{aligned} \tag{σ_9}$$

The result of the application of the above Vadalog rules to the excerpt of the ownership Knowledge Graph is shown in Figure 5. In the rest of the section, we will provide an overview of how KGLM operates to address the two LLM-EKG synergies introduced, namely, *chase-based LLM fine-tuning* and *chase-based explanation retrieval*. Figure 3 illustrates the high-level pipeline of the framework.

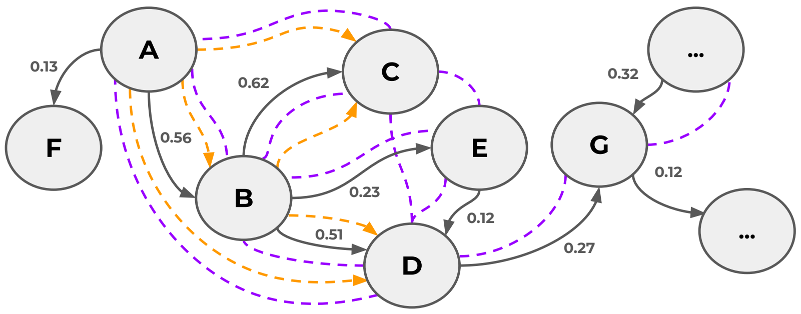


Figure 5 Portion of ownership knowledge graph from Figure 4 augmented with new edges by reasoning on the scenarios from Example 2 and 3. Nodes are companies. Solid edges are *owns* relationships with their shares (*int owns* omitted to avoid clutter). Dashed edges respectively denote *control* relationships (orange, directed) and *close links* (purple, undirected).

3.1 Chase-based LLM Fine-tuning

The goal of this KGLM task is to synthesize question-answering fine-tuning corpora that cover the entire "reasoning space" of the domain of interest, conveying domain-specificity and provenance-awareness to the LLM. Algorithm 1 and Algorithm 2 provide the pseudo-code describing the task, corresponding to the left path of the pipeline in Figure 3.

The first step consists in the execution of the ontological reasoning task of interest with the Vadalog system. Given an extensional knowledge D and the set Σ of Vadalog rules encoding the domain knowledge as the intensional component, the KG is built by augmenting D with the new inferred edges. Consequently, the corresponding chase graph $\Sigma(D)$ is generated (line 2 Algorithm 1). In the presence of results deriving from the application of

Algorithm 1 Reasoning and plan verbalization in KGLM.

```
1:  function ReasoningAndPlanVerbalization($D, \Sigma, glossary$)
2:    $chase \leftarrow$ Vadalog.reason($D, \Sigma$)                          ▷ chase generation
3:    $aggrChase \leftarrow \emptyset$
4:    for each $step$ in $chase$ do                                          ▷ chase aggregation
5:      $stepAggrContrib \leftarrow \emptyset$
6:      if hasAggregate($step$.getRule()) then
7:        $stepAggrContrib \leftarrow$ collectAggrContributors($step, chase$)
8:      $aggrStep \leftarrow$ aggregateStep($step, stepAggrContrib$)
9:      $aggrChase \leftarrow verbChase \cup \{aggrStep\}$
10:   $verbRules \leftarrow$ verbalizeRules($\Sigma, glossary$)               ▷ rules verbalization
11:   $verbPlan \leftarrow$ verbalizePlan($\Sigma$.getLogicPlan(), $verbRules$)
12:   return ($aggrChase, verbPlan$)
```

Algorithm 2 Chase-based LLM fine-tuning task in KGLM.

```
1:  function ChaseBasedFineTuning($aggrChase, verbPlan, model, threshold$)
2:    $tokenizedCorpus \leftarrow$ filter(generate($verbPlan$))             ▷ tokenized corpus generation
3:    $chaseCorpus \leftarrow \emptyset$
4:    for each $aggrStep$ in $aggrChase$ do                                  ▷ chase corpus generation
5:      $chasePromptResp \leftarrow$ map($tokenizedCorpus, aggrStep$)
6:      $chaseCorpus \leftarrow chaseCorpus \cup \{chasePromptResp\}$
7:    for each pair $\langle prompt, resp \rangle$ in $chaseCorpus$ do       ▷ quality-driven optimization
8:      $qualityScore \leftarrow$ checkQuality($\langle prompt, resp \rangle$)
9:      if $qualityScore \leq threshold$ then
10:       $chaseCorpus \leftarrow chaseCorpus \setminus \{\langle prompt, resp \rangle\}$
11:     else
12:       $chaseCorpus \leftarrow chaseCorpus \cup$ paraphrase($\langle prompt, resp \rangle$)
13:   $fineTuningCorpus \leftarrow$ postprocess($chaseCorpus$)
14:   $fineTunedModel \leftarrow$ fineTune($model, fineTuningCorpus$)         ▷ model fine-tuning
15:   return $fineTunedModel$
```

aggregation functions, the CHASE is further processed to collect all the contributors that led to the resulting aggregated value by *unfolding* the corresponding path of CHASE steps altogether [2, 9] (line 9 Algorithm 1).

Here, as previously introduced, our framework uses a *verbalizer* component to transform Σ into a set of natural language sentences via a deterministic transformation. To achieve this, the verbalizer is equipped with a *domain glossary*, containing descriptions of the terms and predicates involved in the rules (line 10 Algorithm 1). In the case of the *company control* scenario introduced in Example 2, the glossary in Figure 6 could be employed.

Predicate	Description
$Owns(x, y, s)$	<x> owns <s> shares of <y>
$ControlledShares(x, z, y, s)$	<x> controls <s> of <y> via <z>
$Control(x, y)$	<x> controls <y>

Figure 6 Domain glossary for the *company control* scenario.

Plan Parts	Verbalized Plan	Tokenized Corpus	Fine-Tuning Corpus
$Owns(x,y,s) \rightarrow ControlledShares(x,y,y,s)$ $\downarrow$ $ControlledShares(x,y,y,s)$, $\text{ts} = \text{sum}(s)$, $\text{ts} > 0.5$ $\rightarrow Control(x,y)$	*Since <x> owns <s> shares of <y>, then <x> controls <s> of <y> via <y>. Since <x> controls <s> of <y> via <y> and <ts> is the sum of <s> and <ts> is higher than 0.5, then <x> controls <y>*	Q1: How does <x> exercise control over <y>? A1: <x> controls <y> as it owns <s> shares, which is the majority	Q1: How does A exercise control over B? A1: A controls B as it owns 0.56 shares, which is the majority
		Q2: How does <x> have a majority over <y>? A2: <x> controls <y> because it owns a total combined number <ts> of its shares, through its subsidiaries	Q2: How does A have a majority over C? A2: A controls C because it owns a total combined number 0.62 of its shares, through its subsidiaries
…		…	…
$IntOwns(x,y,s)$, $s > 0.2$ $\rightarrow CloseLink(x,y)$	*Since <x> has an integrated ownership of <s> over <y>, and <s> is over 0.2, then <x> and <y> are in close link*	Q1: Are <x> and <y> in a close link relationship? A1: <x> and <y> are in a close link relationship, as the integrated ownership <s> of <x> over <y> is higher than 0.2	Q1: Are B and D in a close link relationship? A1: B and D are in a close link relationship, as the integrated ownership 0.51 of B over D is higher than 0.2
…		…	…
$Open(x,y,t_1)$, $\neg\, MarketClosed(t_1)$ $\rightarrow Accepted(x,y,t_1)$ $\downarrow$ $Accepted(x,y,t_1)$, $Price(p_1,t_1)$, $k = y * p_1$ $\rightarrow Position(x,y,k,t_1)$	*Since the trader <x> at time <t_1> sends an order to open a position of size <y> and it is not true that <t_1> is a time when the market is closed, then the order of size <y> by <x> is accepted at time <t_1>. Since the order of size <y> by <x> is accepted and the price is <p_1> at time <t_1>, then <x> holds a position of size <y> and notional <k> at time <t_1>*	Q1: When did <*x*> send an order to open a position with notional <*k*>? A1: The order to open that position was sent at time <t_1>	Q1: When did EGTech send an order to open a position with notional 37.2? A1: The order to open that position was sent at time 1
		Q2: Why was the order sent by trader <*x*> at time <t_1> accepted? A2: Because at time <t_1> the market was open	Q2: Why was the order sent by trader EGTech at time 1 accepted? A2: Because at time 1 the market was open
…		…	…

Figure 7 Generation of domain-specific corpora from scenarios in Examples 1-3 for LLM fine-tuning. Parts of interest of the plan are extracted, verbalized and passed to an LLM for generating the Q&A pair. Then, based on the actual CHASE graph, the final corpus is generated.

The NL translation of the rules leverages the *select-project-join* semantics, rewriting logic-based rules into textual "*since-then* closures". All VADALOG syntactic elements are converted into their textual counterparts. Conjunctions are rendered as *"and"* tokens, built-in operators are represented with specific keywords, e.g., $>$ becomes "*is higher than*", and the same occurs for aggregations, e.g., $x = count(y)$ becomes "x *is the total number of* y", etc. Moreover, rule variables are converted into tokens acting as placeholders. For instance, with respect to Example 2, the rule $Owns(x,y,s) \rightarrow ControlledShares(x,y,y,s)$ (rule σ_4) is verbalized as: "*Since* $<x>$ *owns* $<s>$ *shares of* $<y>$, *then* $<x>$ *controls* $<s>$ *of* $<y>$ *via* $<y>$".

Then, with the verbalization of the rules available, we can generate a fine-tuning corpus to train an LLM to answer novel questions, which might relate to simple information retrieval tasks or more complex ones, i.e., involving an effort toward reasoning on novel input data. To generate the fine-tuning corpus, we exploit the effectiveness of powerful pre-trained LLMs [17], such as GPT-4 and LLAMA 2, in text understanding and manipulation capabilities. In fact, we can ask it to act as a human agent and synthesize a set of possible prompt-response pairs based on an input text. Note that here we have two goals: 1) minimizing the number of "calls" to the LLM, for cost- and time-efficiency reasons; 2) avoiding any ground value (coming from the EKG) being disclosed to the LLM, for data protection reasons. Thus, we leverage the regularity of logical languages and resort to a *lifting technique* based on the creation of a *logic plan* out of Σ (line 11 Algorithm 1). Intuitively, a plan is the equivalent in our context of a database execution plan and can be seen as the dependency graph of the rules of Σ, where nodes represent rules and edges stand for head-body dependencies (similarly, at a high level, to VADALOG's processing pipeline introduced in Section 2). By sending natural language excerpts of the plan, obtained from the aggregation of the verbalized rules, we can automatically generate a corpus of tokenized question-answers, which capture the knowledge encoded in the rules and their interconnections (line 2 Algorithm 2). With such an approach, we achieve a corpus generation pipeline that is cost- and time-effective and that protects confidential data. The tokenized set of prompt-response pairs is then passed on to *filters*, which discard low-quality pairs. For instance, pairs in which the LLM generates new tokens are discarded. Such a step could also resort to a human-in-the-loop approach: in fact, as the pairs are a finite and relatively small set of *templates*, a validator could be hired to discard non-informative ones.

Finally, we materialize the actual fine-tuning corpus via the CHASE graph. For each new fact derived from CHASE procedure, we look up the corresponding verbalized portion of the plan and the tokenized corpus pairs. Each pair is instantiated by mapping the tokens to the corresponding constant arguments of the fact (line 4 Algorithm 2). In Figure 7, we provide some examples of tokenized corpora and their instantiations over artificial data for each of our presented use cases. The corpus undergoes a more thorough quality check where each pair is filtered according to a BERT-score-based scoring model that returns a so-called *R-score*, to evaluate generated text on coherence and factual consistency with the input text [65] (line 7 Algorithm 2). The threshold under which a pair is dropped can be selected by the user. The filtered-in pairs are enhanced via *NLP paraphrasing* to improve generalization, cleansed with additional post-processing procedures, and finally injected into the LLM for domain-specific question-answering fine-tuning (line 14 Algorithm 2).

3.2 Chase-based Explanation Retrieval

The CHASE-based fine-tuning discussed above enables us to inject into the LLM the awareness of the full domain of interest, encapsulating both factual data and the logical connections between them in the form of provenance. However, due to the *closed-book* nature of the approach [57], we empirically observed how the fine-tuned model alone is not able to effectively address complex questions regarding why and how a certain fact exists in the augmented KG. Indeed, answering such *explanatory questions* often involves composing back along the provenance paths that led, from extensional facts, to infer intensional ones in the reasoning process. To address this, we extended KGLM with a dedicated module that supports the LLM in correctly handling such questions by retrieving the CHASE-based explanations and injecting them into the model to enrich its answers. Algorithm 1 and Algorithm 3 provide the pseudo-code describing the task, corresponding to the right path of the pipeline in Figure 3.

The procedure assumes that the first steps, introduced in the context of the LLM fine-tuning task, have already occurred. The reasoning task was performed by VADALOG (line 2 Algorithm 1), the CHASE graph was generated, contributors to aggregations were collected (line 9 Algorithm 1), and the logic plans were translated into combinations of natural language sentences corresponding to the rules in Σ (line 11 Algorithm 1).

Now, let us consider a user interacting with the KG and asking questions about a possible explanation of generated facts. As summarized in Figure 8, this can be performed by either selecting the corresponding edge of the KG or in NL with a prompt-based interaction,

Algorithm 3 Chase-based LLM explanation retrieval task in KGLM.

```
function ChaseBasedExplRetrieval($chase, verbPlan, model, glossary, query$)
    $fact \leftarrow$ extractFact($model, query, glossary$)                ▷ queried fact extraction
    $factStep \leftarrow$ extractStep($chase, fact$)
    $explanationSubgraph \leftarrow \{factStep\}$
    $predecessors \leftarrow \{factStep\}$
    while not $predecessors.isEmpty()$ do                  ▷ explanation subgraph creation
        $currentStep \leftarrow predecessors.dequeue()$
        $currentPreds \leftarrow$ getPredecessors($currentStep, chase$)
        for each $predStep$ in $currentPreds$ do
            if not $predStep$ in $explanationSubgraph$ then
                $explanationSubgraph \leftarrow explanationSubgraph \cup \{predStep\}$
                $predecessors.enqueue(predStep)$
    $detExplanation \leftarrow$ concatenate(map($verbPlan, explanationSubgraph$))
    $refExplanation \leftarrow model.$refine($detExplanation$)
    return $model.$answer($query, refExplanation$)
```

depending on the application KGLM is running into. In both cases, the LLM can extract from the question the corresponding fact of interest, whose explanation is requested by leveraging the model's acquired knowledge of the KG and the glossary (line 2 Algorithm 3).

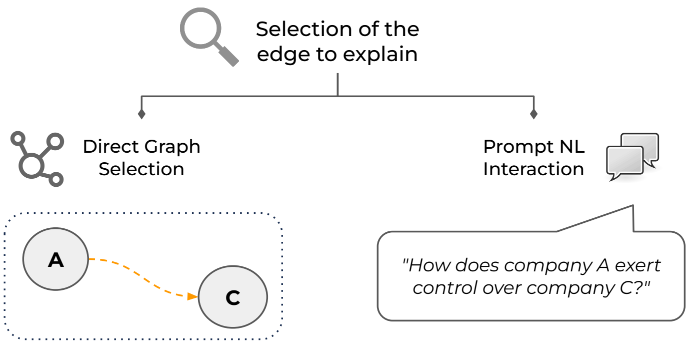

Selection of the edge to explain
Direct Graph Selection
Prompt NL Interaction
A
C
"How does company A exert control over company C?"

Figure 8 Possible interactions with KGLM to select facts to explain.

At this point, the actual retrieval process begins. First, the CHASE step that led to the inference of the fact is identified in the CHASE graph (line 3 Algorithm 3). As we are interested in the full explanation of the fact, we then perform a *back-composition* along the provenance paths (according to the previously mentioned *unfolding* approach [2, 9]), from the fact of interest up to the extensional ones that began such sequences of rule activations during the reasoning process. The result is an *explanation subgraph* of the CHASE graph, featuring both the facts and the activated rules in the paths (line 6 Algorithm 3). For instance, let us consider the portion of ownership KG in Figure 5, augmented with *control* edges derived from the reasoning task. Now, we may wonder *"How does company A exert control over company C?"* and submit such a request to KGLM as an NL prompt-based interaction. First, the LLM identifies the fact "$Control(A, C)$" of interest. Once the fact has been correctly identified, the corresponding trace is retrieved from the CHASE graph, as shown in Figure 9.

Next, we leverage the previously generated verbalized plan, mapping its tokens to the corresponding constant arguments of the facts in the subgraph. By concatenating the resulting verbalized CHASE steps, we obtain the deterministic NL explanation of the fact

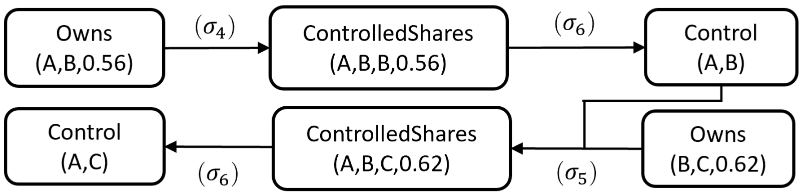


Figure 9 An instance of explanation subgraph for the *company control* scenario.

of interest as a sequence of *"Since {body}, then {head}"* sentences (line 13 Algorithm 3). Note that the final order of the verbalizations reflects the breadth-first traversal of the graph and thus respects logical dependency. However, the explanation produced by the verbalizer can be long, complex, and hard to read, especially in the presence of complex provenance paths. For instance, continuing with our example, we get the following explanation for fact "*Control*(A, C)": *"Since A owns* 0.56 *shares of B, then A controls* 0.56 *of B via B. Since A controls* 0.56 *of B via B and* 0.56 *is higher than* 0.5*, then A controls B. Since A controls B and B owns* 0.62 *shares of C, then A controls* 0.62 *of C via B. Since A controls* 0.62 *of C via B and* 0.62 *is higher than* 0.5 *then A controls C".* To make it more understandable, we leverage the text manipulation capabilities of our model, improving fluency and clarity of the result. By prompting the LLM with the following request: *"Please produce a more readable version of the explanation: ..."*, we achieve a refined report that is highly accurate in content and comprehensible (line 14 Algorithm 3). For instance, the above explanation becomes: *"A's direct ownership of* 0.56 *of B translates to its control over B. With B's ownership of* 0.62 *of C, A, through B, also controls* 0.62 *of C. As the percentage exceeds* 0.5*, A effectively controls C".* Finally, the explanation is passed to our domain-aware model, acting as KG interface, to answer the original user question (line 15 Algorithm 3).

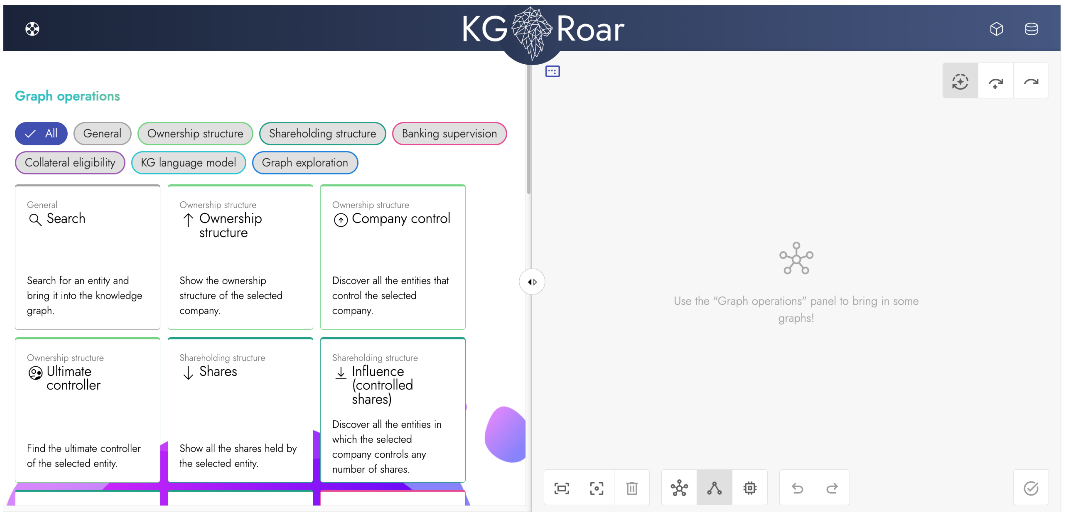
4 KGLM Integration for Financial Applications

We recently introduced KG-Roar [13], a framework we created to showcase ontological reasoning with Vadalog on real-world cases that can be suitably modeled with a KG. It consists of a web-based environment designed for the interactive development and navigation of logical KGs derived from augmenting an input graph database with intensional definitions of new nodes and edges in the form of Vadalog programs. Such programs are user-defined or pre-built code snippets encapsulated in *reasoning widgets* (as illustrated in Figure 10), and act as a metaphor for capturing and integrating business-specific knowledge into the KG, thus enriching its representational power. Indeed, KG-Roar offers an interactive productivity environment where the user can select widgets of interest and augment the KG interactively at runtime with new knowledge thanks to the scalability and responsiveness of the underlying Vadalog reasoning engine. A *virtual* representation of the KG is provided in an interactive visualization window within the environment. The goal of KG-Roar is to enable users to perform complex analyses, seamlessly building, browsing, and querying even complex and large knowledge graphs, such as the European ownership KG of financial companies.

KGLM in KG-Roar. To support such a purpose, we enrich the KG-Roar environment with the KGLM framework presented in the previous section. Going beyond the visual representation of the derived edges in the KG, often unsatisfactory as missing the actual motivation for their existence (i.e., their provenance), we provide users with a *knowledge pal* that enables natural language-based interactions with the underlying KG. Specifically, such an agent acts as a chatbot, standing upon LLAMA 2-70B models that are specialized in the domain of interest by leveraging KGLM's CHASE-based fine-tuning corpora. Depending on the nature of the user question, it performs distinct actions to provide the answer.

- In case of *descriptive queries*, i.e., questions that are focused on investigating the specifics of a certain fact in the KG, the knowledge pal leverages the expertise of the domain, acquired via fine-tuning, and the verbalized rules provided in a RAG-like mechanism to generate the answer. For instance, in the context of Example 1, to the question *"What is the notional of EGTech's position at time 1?"*, it responds with *"EGTech's notional at time 1 is 37.2\$, equal to the product between 124\$, the price of the asset at that time, and 0.3, the size of the opened position"*.
- In case of *explanatory queries*, i.e., more complex questions that are focused on investigating why a certain fact exists in the KG (that is, its provenance), the knowledge pal retrieves the full CHASE-based explanation of the fact from KGLM, further enhancing its readability and providing it to the user as an accurate business report. For instance, in the context of Example 1, to the question *"How does EGTech make profits from its trading activity?"*, it responds with *"By opening a position of size 0.3 at time 1, having the position accepted with an initial price of 124\$ and a notional of 37.2\$ at time 1, and then closing it at time 9 with a final price of 147\$, thus achieving a profit of 6.9\$"*.

Let us consider the application of the *company control* scenario from Example 2 and the *close link* one from Example 3 to showcase KGLM's integration into KG-ROAR. As previously mentioned, KGLM supports the selection of the fact to explain from the corresponding edge in the knowledge graph. Thus, it is possible to click on an edge in the visualized KG to request the corresponding explanation, as illustrated in Figure 11. In the presence of highly connected KGs, such as the one in Figure 12, the visual exploration becomes less intuitive and the request can directly be prompted in NL to the knowledge pal. In both cases, the generated response will be presented in a dedicated box that can be interactively explored by highlighting tagged entities both in the text and in the visualized KG, and KG-ROAR will zoom in on the portion of the graph involved.



General
Search
Search for an entity and bring it into the knowledge graph.

Ownership structure
Ownership structure
Show the ownership structure of the selected company.

Ownership structure
Company control
Discover all the entities that control the selected company.

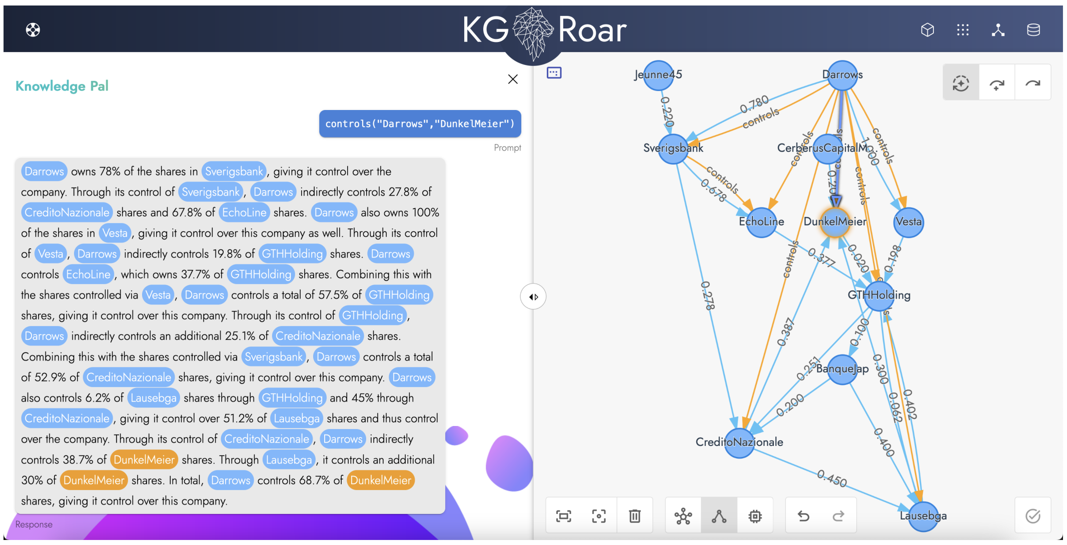
Ownership structure
Ultimate controller
Find the ultimate controller of the selected entity.

Shareholding structure
Shares
Show all the shares held by the selected entity.

Shareholding structure
Influence (controlled shares)
Discover all the entities in which the selected company controls any number of shares.

Use the "Graph operations" panel to bring in some graphs!

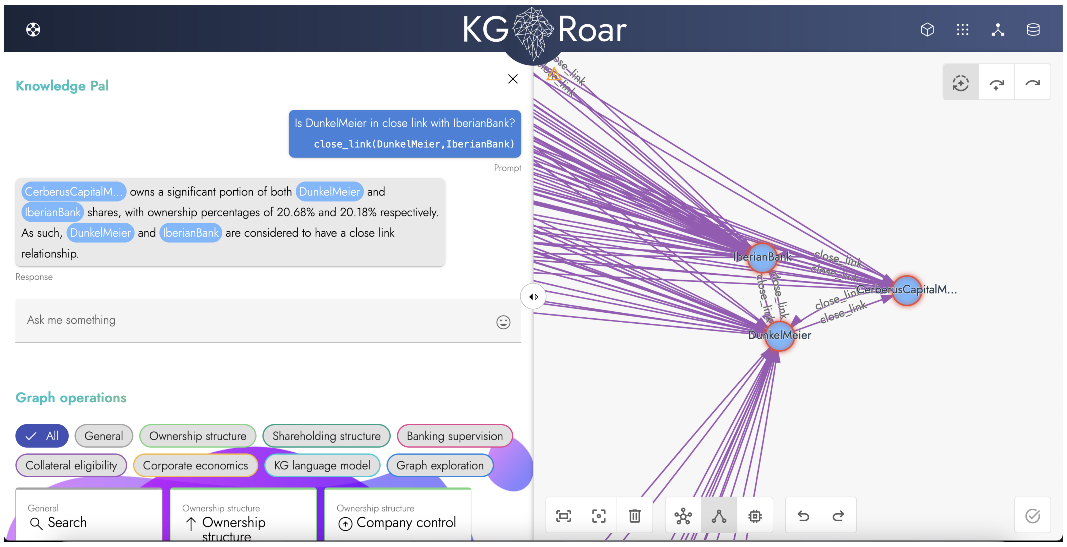
Figure 10 Financial use cases in the form of executable widgets.



```
controls("Darrows","DunkelMeier")
```

Darrows owns 78% of the shares in Sverigsbank, giving it control over the company. Through its control of Sverigsbank, Darrows indirectly controls 27.8% of CreditoNazionale shares and 67.8% of EchoLine shares. Darrows also owns 100% of the shares in Vesta, giving it control over this company as well. Through its control of Vesta, Darrows indirectly controls 19.8% of GTHHolding shares. Darrows controls EchoLine, which owns 37.7% of GTHHolding shares. Combining this with the shares controlled via Vesta, Darrows controls a total of 57.5% of GTHHolding shares, giving it control over this company. Through its control of GTHHolding, Darrows indirectly controls an additional 25.1% of CreditoNazionale shares. Combining this with the shares controlled via Sverigsbank, Darrows controls a total of 52.9% of CreditoNazionale shares, giving it control over this company. Darrows also controls 6.2% of Lausebga shares through GTHHolding and 45% through CreditoNazionale, giving it control over 51.2% of Lausebga shares and thus control over the company. Through its control of CreditoNazionale, Darrows indirectly controls 38.7% of DunkelMeier shares. Through Lausebga, it controls an additional 30% of DunkelMeier shares. In total, Darrows controls 68.7% of DunkelMeier shares, giving it control over this company.

Figure 11 Instance of explanation panel for *company control* use case via direct graph selection.



Is DunkelMeier in close link with IberianBank?

```
close_link(DunkelMeier,IberianBank)
```

CerberusCapitalM... owns a significant portion of both DunkelMeier and IberianBank shares, with ownership percentages of 20.68% and 20.18% respectively. As such, DunkelMeier and IberianBank are considered to have a close link relationship.

Figure 12 Instance of explanation panel for *close link* use case via prompt NL interaction.

5 Conclusion

In today's industrial landscape, we observe an increasingly pressing demand for AI applications to sustain transparency, fairness, and accountability of decision-making processes, especially in high-stakes domains like finance and bio-medicine. To achieve this, novel neuro-symbolic solutions are rising to bridge the gap between the opaque nature of ML-based technologies such as LLMs and the required human-interpretable provenance of their responses.

In this paper, we strived to contribute to such a paradigm shift by presenting KGLM, the first, to the best of our knowledge, neuro-symbolic pipeline that enhances LLMs with domain-specific knowledge and provenance-based explainability by leveraging the inherent domain-awareness of Knowledge Representation and Reasoning methodologies at the foundation of the Vadalog system. Capitalizing on our experience in the financial field, we built a framework that can act as a natural language interface to Enterprise Knowledge Graphs, empowering business analysts to investigate the rationale behind data investigation performed via reasoning tasks in a human-oriented fashion. We also integrated our solution into a well-established KG environment for financial tasks, KG-Roar, which allows such users to conduct graph-based data analysis, now enhanced by the possibility of interacting with them in natural language. By further leveraging provenance information, we also aim to expand our KGLM framework in future works, injecting reasoning capabilities into Large Language Models in a chain-of-thought fashion to directly reason over EKGs, possibly enriched by additional unstructured data. We believe that such contributions could become significant assets for organizations, enabling them to harness the advanced capabilities of LLMs without sacrificing the clarity and accountability of informed decision-making processes.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

2 Foto N. Afrati, Manolis Gergatsoulis, and Francesca Toni. Linearisability on datalog programs. *Theor. Comput. Sci.*, 308(1-3):199–226, 2003. doi:10.1016/S0304-3975(02)00730-2.

3 Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. Augmenting logic-based knowledge graphs: The case of company graphs. In *KR4L@ECAI*, volume 3020, pages 22–27. CEUR-WS.org, 2020.

4 Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. Weaving enterprise knowledge graphs: The case of company ownership graphs. In *EDBT*, pages 555–566. OpenProceedings.org, 2020.

5 Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. Fine-tuning large enterprise language models via ontological reasoning. In *International Joint Conference on Rules and Reasoning*, pages 86–94. Springer, 2023.

6 Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. "please, vadalog, tell me why": Interactive explanation of datalog-based reasoning. In Letizia Tanca, Qiong Luo, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani, editors, *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 834–837. OpenProceedings.org, 2024. doi:10.48786/EDBT.2024.82.

7 Teodoro Baldazzi, Luigi Bellomarini, Marco Favorito, and Emanuel Sallinger. Ontological reasoning over shy and warded datalog+/–for streaming-based architectures. In *International Symposium on Practical Aspects of Declarative Languages*, pages 169–185. Springer, 2024.

8 Teodoro Baldazzi, Luigi Bellomarini, Markus Gerschberger, Aditya Jami, Davide Magnanimi, Markus Nissl, Aleksandar Pavlović, and Emanuel Sallinger. Vadalog: Overview, extensions and business applications. *Reasoning Web. Causality, Explanations and Declarative Knowledge: 18th International Summer School 2022, Berlin, Germany, September 27–30, 2022, Tutorial Lectures*, pages 161–198, 2023.

9 Teodoro Baldazzi, Luigi Bellomarini, Emanuel Sallinger, and Paolo Atzeni. Eliminating harmful joins in warded datalog+/-. In *RuleML+RR*, volume 12851 of *Lecture Notes in Computer Science*, pages 267–275. Springer, 2021.

10 Teodoro Baldazzi, Davide Benedetto, Matteo Brandetti, Adriano Vlad, Luigi Bellomarini, and Emanuel Sallinger. Datalog-based reasoning with heuristics over knowledge graphs. In *Datalog*, volume 3203 of *CEUR Workshop Proceedings*, pages 114–126. CEUR-WS.org, 2022.

11 Pablo Barceló and Reinhard Pichler. *Datalog in Academia and Industry: Second International Workshop, Datalog 2.0, Vienna, Austria, September 11-13, 2012, Proceedings*, volume 7494. Springer, 2012.

12 Luigi Bellomarini, Lorenzo Bencivelli, Claudia Biancotti, Livia Blasi, Francesco Paolo Conteduca, Andrea Gentili, Rosario Laurendi, Davide Magnanimi, Michele Savini Zangrandi, Flavia

Tonelli, Stefano Ceri, Davide Benedetto, Markus Nissl, and Emanuel Sallinger. Reasoning on company takeovers: From tactic to strategy. *Data Knowl. Eng.*, 141:102073, 2022.

13 Luigi Bellomarini, Marco Benedetti, Andrea Gentili, Davide Magnanimi, and Emanuel Sallinger. Kg-roar: Interactive datalog-based reasoning on virtual knowledge graphs. *Proc. VLDB Endow.*, 16(12):4014–4017, August 2023.

14 Luigi Bellomarini, Davide Benedetto, Georg Gottlob, and Emanuel Sallinger. Vadalog: A modern architecture for automated reasoning with large knowledge graphs. *Inf. Syst.*, 105:101528, 2022. `doi:10.1016/j.is.2020.101528`.

15 Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge graphs and enterprise AI: the promise of an enabling technology. In *ICDE*, pages 26–37, 2019.

16 Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *Proc. VLDB Endow.*, 11(9):975–987, May 2018. `doi:10.14778/3213880.3213888`.

17 Tom Brown and et al. Language models are few-shot learners. In *NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

18 Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 316–330. Springer Berlin Heidelberg, 2001.

19 Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. A system for explainable answer set programming. *Electronic Proceedings in Theoretical Computer Science*, 325:124–136, 2020. `doi:10.4204/eptcs.325.19`.

20 Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14:57–83, 2012. `doi:10.1016/j.websem.2012.03.001`.

21 Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *2010 25th annual IEEE symposium on logic in computer science*, pages 228–242. IEEE, 2010.

22 Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.

23 Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *ACM SIGMOD 2006*, pages 745–747, 2006.

24 Luciano Caroprese, Eugenio Vocaturo, and Ester Zumpano. Argumentation approaches for explanaible ai in medical informatics. *Intelligent Systems with Applications*, 16:200109, 2022. `doi:10.1016/j.iswa.2022.200109`.

25 Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166, 1989.

26 James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474, 2009.

27 Sarah Cohen-Boulakia, Olivier Biton, Shirley Cohen, and Susan Davidson. Addressing the provenance challenge using zoom. *Concurrency and Computation: Practice and Experience*, 20(5):497–506, 2008.

28 Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, September 2001. `doi:10.1145/502807.502810`.

29 Daniel Deutch, Nave Frost, and Amir Gilad. Provenance for natural language queries. *Proc. VLDB Endow.*, 10(5):577–588, January 2017. `doi:10.14778/3055540.3055550`.

30 Daniel Deutch, Nave Frost, and Amir Gilad. Natural language explanations for query results. *SIGMOD Rec.*, 47(1):42–49, September 2018. `doi:10.1145/3277006.3277017`.

31 Xin Luna Dong. Generations of knowledge graphs: The crazy ideas and the business impact. *arXiv preprint arXiv:2308.14217*, 2023.

32 Owen P. Dwyer, Teodoro Baldazzi, Jim Davies, Emanuel Sallinger, and Adriano Vlad. Reasoning over health records with vadalog: a rule-based approach to patient pathways. In Jan Vanthienen, Tomás Kliegr, Paul Fodor, Davide Lanti, Dörthe Arndt, Egor V. Kostylev, Theodoros Mitsikas, and Ahmet Soylu, editors, *Proceedings of the 17th International Rule Challenge and 7th Doctoral Consortium @ RuleML+RR 2023 co-located with 19th Reasoning Web Summer School (RW 2023) and 15th DecisionCAMP 2023 as part of Declarative AI 2023, Oslo, Norway, 18 - 20 September, 2023*, volume 3485 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL: `https://ceur-ws.org/Vol-3485/paper9111.pdf`.

33 Esra Erdem and Umut Oztok. Generating explanations for biomedical queries. *Theory and Practice of Logic Programming*, 15(1):35–78, 2015.

34 European Central Bank. Guideline (eu) 2011/14 of the ecb guideline, 2011.

35 Jorge Fandinno and Claudia Schulz. Answering the "why" in answer set programming–a survey of explanation approaches. *Theory and Practice of Logic Programming*, 19(2):114–203, 2019.

36 Boris Glavic, Renée J. Miller, and Gustavo Alonso. *Using SQL for Efficient Generation and Querying of Provenance Information*, pages 291–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. `doi:10.1007/978-3-642-41660-6_16`.

37 Goetz Graefe and William J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218. IEEE Computer Society, 1993.

38 Todd J Green, Shan Shan Huang, Boon Thau Loo, Wenchao Zhou, et al. Datalog and recursive query processing. *Foundations and Trends® in Databases*, 5(2):105–195, 2013.

39 Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

40 Todd J. Green and Val Tannen. The semiring framework for database provenance. In *PODS*, pages 93–99. ACM, 2017.

41 A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, and L. Bellomarini. Distributed company control in company shareholding graphs. In *IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2637–2648, Los Alamitos, CA, USA, 2021.

42 Kyle Hamilton, Aparna Nayak, Bojan Bozic, and Luca Longo. Is neuro-symbolic AI meeting its promise in natural language processing? A structured review. *CoRR*, abs/2202.12205, 2022. `arXiv:2202.12205`.

43 Melanie Herschel and Marcel Hlawatsch. Provenance: On and behind the screens. In *SIGMOD 2016*, pages 2213–2217, New York, NY, USA, 2016. Association for Computing Machinery. `doi:10.1145/2882903.2912568`.

44 Jason I. Hong. Teaching the fate community about privacy. *Commun. ACM*, 66(8):10–11, July 2023.

45 David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

46 Dominik K Kanbach, Louisa Heiduk, Georg Blueher, Maximilian Schreiter, and Alexander Lahmann. The genai is out of the bottle: generative artificial intelligence from a business model innovation perspective. *Review of Managerial Science*, pages 1–32, 2023.

47 David Koop, Marta Mattoso, and Juliana Freire. *Provenance in Workflows*, pages 2912–2916. Springer New York, New York, NY, 2018. `doi:10.1007/978-1-4614-8265-9_80745`.

48 Markus Krötzsch and Veronika Thost. Ontologies for knowledge graphs: Breaking the rules. In *International Semantic Web Conference*, pages 376–392. Springer, 2016.

49 Seokki Lee, Bertram Ludäscher, and Boris Glavic. Provenance summaries for answers and non-answers. *Proc. VLDB Endow.*, 11(12):1954–1957, August 2018. `doi:10.14778/3229863.3236233`.

50 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

51 Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. In *IJCAI 2020*, 2021.

52 Davide Magnanimi and Michela Iezzi. Ownership graphs and reasoning in corporate economics. In *EDBT/ICDT Workshops*, volume 3135 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

53 David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM TODS*, 4(4):455–469, 1979. `doi:10.1145/320107.320115`.

54 Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*, 2023.

55 Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. ., 2018.

56 Yann Ramusat, Silviu Maniu, and Pierre Senellart. Semiring provenance over graph databases. In *Proceedings of the 10th USENIX Conference on Theory and Practice of Provenance*, TaPP'18, page 7, USA, 2018. USENIX Association.

57 Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *EMNLP (1)*, pages 5418–5426. ACL, 2020.

58 Sudeepa Roy and Dan Suciu. A formal approach to finding explanations for database queries. In *SIGMOD 2014*, pages 1579–1590, New York, NY, USA, 2014. Association for Computing Machinery. `doi:10.1145/2588555.2588578`.

59 Pierre Senellart. *Provenance in Databases: Principles And Applications*, pages 104–109. Springer-Verlag, Berlin, Heidelberg, 2022. `doi:10.1007/978-3-030-31423-1_3`.

60 Yogesh L Simmhan, Beth Plale, and Dennis Gannon. Karma2: Provenance management for data-driven workflows. *International Journal of Web Services Research (IJWSR)*, 5(2):1–22, 2008.

61 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

62 Adriano Vlad, Sahar Vahdati, Mojtaba Nayyeri, Luigi Bellomarini, and Emanuel Sallinger. Towards hybrid logic-based and embedding-based reasoning on financial knowledge graphs. In *EDBT/ICDT Workshops*, volume 3135, 2022.

63 Yisu Remy Wang, Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, and Dan Suciu. Optimizing recursive queries with progam synthesis. In *SIGMOD 2022*, pages 79–93, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3514221.3517827`.

64 Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564, 2023.

65 Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL: `https://openreview.net/forum?id=SkeHuCVFDr`.

66 Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *arXiv preprint arXiv:2309.01029*, 2023.

Traversal-Invariant Characterizations of Logarithmic Space

Siddharth Bhaskar[1]
Department of Computer Science, James Madison University, Harrisonburg, VA, USA

Steven Lindell
Department of Computer Science, Haverford College, Haverford, PA, USA

Scott Weinstein
Department of Philosophy, University of Pennsylvania, Philadelphia, PA, USA

Abstract

We give a novel descriptive-complexity theoretic characterization of L and NL computable queries over finite structures using *traversal invariance.* We summarize this as (N)L = FO + (breadth-first) traversal-invariance.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases Model theory, finite model theory, descriptive complexity theory, logarithmic space, graph traversals

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.2

Related Version *Full Version*: `https://arxiv.org/abs/2006.07067`

1 Presentation invariance

A common phenomenon in mathematics is that some property or quantity is defined in terms of some additional structure, but ends up being invariant of it. Dimension of a vector space and Euler characteristic of a manifold are important examples of this phenomenon; they are defined in terms of a given basis or simplicial complex respectively, but are invariant of the particular one chosen.

This state of affairs is very common in descriptive complexity theory. For example, suppose we want to compute the parity of a given finite set X. If we are given some linear ordering $(X, <)$, there is an inductive program computing the parity of X, but the result computed is independent of the particular ordering. Therefore, we call parity *order-invariant LFP*: computable by an LFP program with a given order, but independent of the specific choice.

The celebrated result of Immerman and Vardi [6, 11] that LFP logic captures polynomial-time queries over families of ordered finite structures can be recast as, *order-invariant* LFP logic captures polynomial-time queries over *all* finite structures. Since then, a wide array of correspondences have been identified between known complexity classes on one hand, and invariant forms of LFP, MSO, or first-order logic on the other. For example, first-order logic and LFP logic invariant in arbitrary numerical predicates captures AC^0 and $\mathrm{P}/_{\mathrm{poly}}$ respectively [9].

Our contribution. We give a novel characterization of logarithmic and nondeterministic logarithmic space queries using presentation-invariant first-order definability (Theorems 28 and 29). The presentations in question are traversals and breadth-first traversals respectively, which are certain types of linear orders on finite graphs.

[1] Corresponding author

This is to our knowledge the first characterization of L or NL that does not rely on any sort of recursion or sequential computation, however limited, such as a function algebra, fixed-point logic, programming language, or automaton.

We find it fascinating and mysterious that passing from traversals to breadth-first traversals in the presentation causes a jump from L to NL in definability power. It begs the question, what other complexity classes can be characterized by certain types of graph search?

Structure of this paper. In Section 2, we discuss traversal- and breadth-first traversal-invariant definability, and show the definability of undirected and directed reachability respectively. In Section 3, we present descriptive-theoretic characterizations of L and NL (Theorems 28 and 29).

Preliminaries and notation. We assume familiarity with basic graph theory, automata theory, and model theory, including the notion of *interpretation*. We will denote graphs and other first-order structures by uppercase Roman letters. By "graph" we always mean "undirected graph;" we will say "directed graph" when we need to. We denote families of structures in a common signature by captial calligraphic letters, e.g., $\mathcal{K}$.

2 Traversals

Traversals are absolutely fundamental in computer science. They give us systematic ways of exploring a finite graph or other sort of network, and lie at the foundation of all sorts of sophisticated algorithms and techniques. Let us isolate the simplest possible version, which we call *generic graph search*, and which operates over a finite nonempty graph G.

1. Initialize a set S to some vertex in G, and repeat the following until $G \setminus S$ is empty.
2. If there is some vertex in the boundary of S, add it to S. Otherwise, add any element of $G \setminus S$ to S.

Generic graph search is nondeterministic, insofar as it does not specify which vertex to add to S. Important refinements of this algorithm include *breadth-first* and *depth-first* search, which specify additional heuristics for how to add vertices to S, without being fully deterministic.

In common parlance, the word *traversal* can refer either to the algorithm or the linear orders of G they produce, but in the current work we reserve the term "traversal," "breadth-first traversal," and "depth-first traversal" for the latter. In this paper, we do not work with depth-first traversals, but we will come back to them in the last section.

▶ **Definition 1.** *For a finite graph G, $(G, <)$ is a* traversal *(resp. breadth-first traversal, depth-first traversal) in case some instance of generic graph search (resp. breadth-first search, depth-first search) of G visits its vertices in order $<$.*

Corneil and Krueger [2] discovered that, in fact, these traversals are first-order definable in the language of ordered graphs.

▶ **Lemma 2.** *For any finite graph G and linear ordering $<$ of its vertices*

- *$(G, <)$ is a traversal iff*
$$(G, <) \models (\forall u < v < w)(uEw \to (\exists x < v)\, xEv),$$
- *$(G, <)$ is a breadth-first traversal iff*
$$(G, <) \models (\forall u < v < w)(uEw \to (\exists x < v)\, x \le u \wedge xEv),$$
- *and $(G, <)$ is a depth-first traversal iff*
$$(G, <) \models (\forall u < v < w)(uEw \to (\exists x < v)\, x \ge u \wedge xEv).$$

Note that connected components of G induce intervals in a traversal. Notice also how the definitions of breadth-first traversal and depth-first traversal refine the notion of traversal in opposing ways: given a vertex v that occurs between two endpoints u and w of a single edge, v must have some prior neighbor in a plain traversal. In a breadth-first traversal, there must be some prior neighbor less than or equal to u, and in a depth-first traversal, there must be some prior neighbor greater than or equal to u.

It is an easy but important fact that

▶ **Lemma 3.** *Every finite graph admits a traversal; a fortiori, every finite graph admits both a breadth-first traversal and a depth-first traversal.*

In the present paper we characterize L and NL using traversals and breadth-first traversals respectively; it is an open question whether depth-first traversals similarly characterize some complexity class.

2.1 Traversal-invariant definability

We now present the fundamental definability-theoretic concepts in this paper. We use the standard model-theoretic notion of an *interpretation* in this definition; for details see the Appendix. If $\mathcal{K}$ is some family of structures in a common signature, by a "query over $\mathcal{K}$," we mean a boolean query, i.e., an isomorphism-closed subset of $\mathcal{K}$.[2]

▶ **Definition 4.** *Suppose $K \subseteq K^+$ are signatures, $\mathcal{K}$ is a nonempty family of K-structures, and $\mathcal{P}$ is a nonempty family of K^+-structures, such that for any $A \in \mathcal{P}$, its K-reduct $A|_K$ is in $\mathcal{K}$.*

A first-order sentence φ over $\mathcal{P}$ is $(\mathcal{K}, \mathcal{P})$-invariant *in case for any two structures A and B in $\mathcal{P}$ with the same domain, if $A|_K \cong B|_K$, then $A \models \varphi \iff B \models \varphi$.*

▶ **Definition 5.** *An n-*pointed graph *is a graph expanded with n constants. Let Γ_n be the language of n-pointed graphs, i.e., a binary relation symbol and n constant symbols.*

▶ **Definition 6.** *Let $\mathcal{G}'$ be a family of finite n-pointed graphs, $\mathcal{T}$ be the set of all expansions of structures in $\mathcal{G}'$ by any traversal, and φ be a $(\mathcal{G}', \mathcal{T})$-invariant sentence. Then for any $G \in \mathcal{G}'$, we write*

$$G \models (\mathfrak{T} <)\, \varphi$$

to indicate that for some (equivalently, any) traversal $<$ of G, $(G, <) \models \varphi$. Similarly, we write

$$G \models (\mathfrak{B} <)\, \varphi$$

if φ is $(\mathcal{G}', \mathcal{B})$-invariant where $\mathcal{B}$ is the set of all expansions by breadth-first traversals.

▶ **Definition 7.** *Let K be a signature, $\mathcal{K}$ be some family of K-structures and Q a query over $\mathcal{K}$. We say that Q is* basic traversal-invariant definable *if there exists some $n \in \mathbb{N}$, a family of finite n-pointed graphs $\mathcal{G}'$, a $(\mathcal{G}', \mathcal{T})$-invariant sentence φ, and an interpretation $\pi : \Gamma_n \to K$, such that*

[2] We will represent n-ary queries over $\mathcal{K}$ by boolean queries over the family of structures obtained by expanding every structure in $\mathcal{K}$ by any n points.

1. *π is an interpretation $\mathcal{K}$ to $\mathcal{G}'$, and*
2. *for any $A \in \mathcal{K}$,*

$$A \in Q \iff A^\pi \models (\mathfrak{T} <)\, \varphi,$$

where $\mathcal{T}$ is the set of all expansions by a traversal of all graphs in $\mathcal{G}'$.[3]

We define basic breadth-first traversal (BFT)-invariant definable *similarly. We also write $A \models ((\mathfrak{T} <)\, \varphi)^\pi$ to mean $A^\pi \models (\mathfrak{T} <)\, \varphi$.*

Note that in our definition of traversal- or BFT-invariant definability, $\mathcal{G}'$ is not required to be the family of *all* finite n-pointed graphs, though it typically will be. Note also that $\mathcal{K}$ must be a family of finite structures if there is to be a interpretation $\pi : \mathcal{K} \to \mathcal{G}'$.

▶ **Definition 8.** *Let K be a signature, $\mathcal{K}$ be some family of K-structures and Q a query over $\mathcal{K}$. Then Q is* traversal-invariant definable *(resp.* BFT-invariant definable*) if it is a boolean combination of basic traversal-invariant (resp. basic BFT-invariant) definable queries.*

We collect some important examples:

▶ **Lemma 9.** *The following queries are traversal-invariant definable over the indicated families of structures $\mathcal{K}$:*
1. *Undirected st-connectivity, over all finite 2-pointed graphs.*
2. *The family of all acyclic graphs, over all finite graphs.*
3. *The family of all bipartite graphs, over all finite graphs.*
4. *The family of even-sized finite linear orders, over all finite linear orders.*

Proof. Let $\mathcal{G}_\in$ be the family of all finite 2-pointed graphs with constants s and t. The binary reachability relation is actually definable by a single $(\mathcal{G}_\in, \mathcal{T})$-invariant sentence, which says that there is no w with no prior neighbor such that $s < w \leq t$ or $t < w \leq s$. Since components of G induce intervals in $(G, <)$, this formula asserts there is no interval separating s and t into separate connected components.

Acyclicity is similarly the spectrum of a $(\mathcal{G}, \mathcal{T})$-invariant sentence. A graph is acyclic iff, relative to any traversal, no vertex has two or more prior neighbors.

The *square* of a graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$, where $E^2(x, y)$ iff x and y are connected by a path of length exactly two. Then G is bipartite iff G^2 is disconnected. Since G^2 is definable as a translation of G under an interpretation $\pi : \mathcal{G} \to \mathcal{G}$, and since connectivity is traversal-invariant definable, so is bipartiteness.

The parity of a linear order is also equivalent to the connectivity of a translation. Specifically, connect u and v by an edge iff $u = v \pm 2 \mod n$, where n is the size of the order. Then the resulting graph is either a single cycle or a union of two cycles depending on whether n is odd or even respectively. ◀

Since, e.g., connectivity and acyclicity are not Gaifman-local queries [3, 9], it follows that traversal-invariance is strictly more expressive than order-invariance.

[3] See the Appendix for the definition of notions and notations involving interpretations.

2.2 Directed reachability

Here we deal with the question of *directed st*-connectivity using BFT invariance. In fact, we will need something more than the directed graph structure, but the result will be invariant of it, an apparent asymmetry with the undirected case that will be resolved in the next section.

This construction is substantially more sophisticated than our examples above. We reduce directed reachability to an equidistance problem over undirected graphs, which we solve with the appropriate BFT-invariant sentence.

▶ **Definition 10.** *If A is a finite structure, we say that a* successor expansion (A, S) *of A is a structure of the form $(A, \min, \max, S)$, where* min *and* max *are constants and S is a* successor function *on a total order with endpoints* min *and* max*.*

If K is a signature, let (K, S) be the signature of successor expansions of K-structures.[4] *If $\mathcal{K}$ is a family of finite K-structures, let $\mathcal{K}^{\mathcal{S}}$ be the family of all successor expansions of structures in $\mathcal{K}$.*

▶ **Definition 11** (Successor Invariance)**.** *For any family $\mathcal{K}$ of finite structures, let $\mathcal{K}^{\mathcal{S}}$ be the set of all successor expansions of $\mathcal{K}$. A query Q over $\mathcal{K}^{\mathcal{S}}$ is* successor-invariant *in case for any $A, B \in \mathcal{K}^{\mathcal{S}}$, if $A|_K \cong B|_K$, then $A \models Q \iff B \models Q$.*

For any $C \in \mathcal{K}$, we say $C \models (\mathfrak{S}S)\, Q$ iff some (equivalently, any) successor expansion of C satisfies Q.

▶ **Definition 12.** *Let $\mathcal{D}_n$ be the family of all finite n-pointed directed graphs, and $\mathcal{G}_n$ be the family of all finite n-pointed graphs. Let $\mathcal{D}_n^{\mathcal{S}}$ be the family of all successor expansions of all finite directed n-pointed graphs.*

The interpretation ρ. We present an interpretation defined in [10] that translates directed successor graphs into undirected graphs. Let (x, y, z) be the constants of Γ_3 and (s, t) be the constants of (Γ_2, S). Consider the binary interpretation $\rho : \Gamma_3 \to (\Gamma_2, S)$ defined by

$$E^\rho(u, a; v, b) \equiv \big(S(a) = b \wedge E'(u, v)\big) \vee \big(S(b) = a \wedge E'(v, u)\big).$$

$$x^\rho = (s, \min)$$

$$y^\rho = (s, \max)$$

$$z^\rho = (t, \max),$$

where $E'(u, v)$ abbreviates $E(u, v) \vee u = v$. Then ρ is an interpretation $\mathcal{D}_2^{\mathcal{S}} \to \mathcal{G}_3$, because the predicate E^ρ is visibly symmetric. Note that ρ is also quantifier-free. We can express the st reachability problem on $D \in \mathcal{D}_2^{\mathcal{S}}$ into an equidistance problem on D^ρ, cf. Figure 1 for an example. A proof of the following lemma appears in [10] and is deferred to the Appendix.

▶ **Lemma 13.** *For any graph $D \in \mathcal{D}_2^{\mathcal{S}}$, there is a directed path from s to t in D iff the vertices y and z are equidistant from x in D^ρ. Even stronger, if there is no directed path from s to t in D, then either $d(x, z)$ is undefined or $|d(x, y) - d(x, z)| \geq 2$, where d indicates distance in D^ρ.*

[4] Note that there is no symbol for the order with respect to which S is a successor function in the signature (K, S).

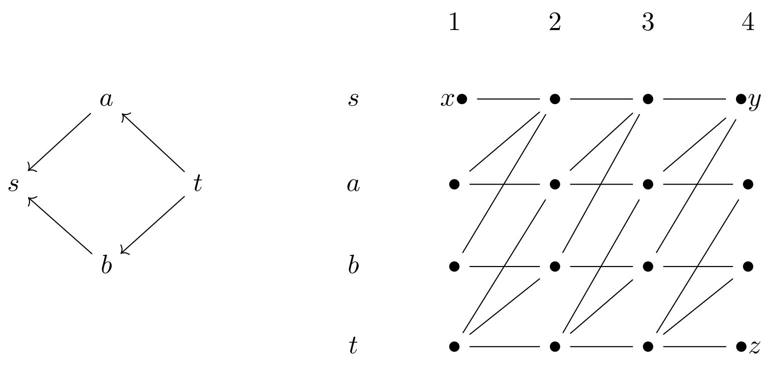


Figure 1 A directed graph to the left, with its undirected ρ-translation to the right. There is an edge between (u, i) and $(v, i + 1)$ on the right exactly when $u = v$ or there is a directed edge $u \to v$ on the left. Consequently, there is a directed path $s \rightsquigarrow t$ on the left exactly when y and z are equidistant from x on the right. Here there is no such path $s \rightsquigarrow t$, $d(x, y) = 3$, and $d(x, z) = 7$.

▶ **Definition 14.** *Let $\mathcal{G}'_3$ be the family of finite 3-pointed undirected graphs with constants x, y, z such that x and y lie in the same connected component and, if z does as well, then $|d(x, y) - d(x, z)| \neq 1$.*

By Lemma 13, ρ is in fact an interpretation $\mathcal{D}^S_2 \to \mathcal{G}'_3$.

Breadth-first traversals and quasi-levels. On a graph with a distinguished source for each connected component, vertices are naturally partitioned into *levels* according to their distance from their respective source. If we fix a BFT of a graph, and let the source of each connected component be its least element, then the resulting levels induce intervals in that traversal.[5] Moreover, every edge of the graph is either within levels or between adjacent levels. The least neighbor of every vertex (except the source) is in the previous level.

It is probably impossible to recognize when two nodes are in the same level using first-order logic on graphs, even given a BFT. However we can do almost as well.

▶ **Definition 15.** *Let $(V, E, <)$ be a finite graph expanded by a breadth-first traversal. A* quasi-level *is a nonempty interval I of (V, E) such that $w \in I \iff p(w) < v \leq w$, where v is the least element of I and $p(w)$ the least neighbor of w (cf. Figure 2).*

Observe that it is easy to define when two vertices v and w occur in a common quasi-level, by the formula

$$(p(w) < v \leq w) \vee (p(v) < w \leq v).$$

Notice that if two vertices occur in a common quasi-level, then their distances from their (necessarily common) source cannot differ by more than 1. If two vertices occur in no common quasi-level, then either they are in different connected components, or the distances from their common source cannot be equal.

[5] Recall that connected components induce intervals of a traversal, so it suffices to observe that levels induce intervals within connected components.

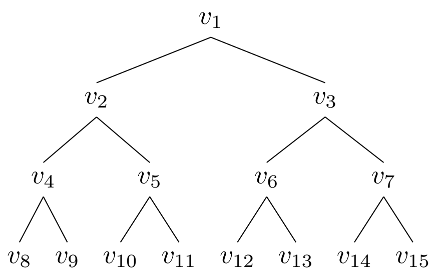


Figure 2 $\{v_2, v_3\}$, $\{v_3, v_4, v_5\}$, $\{v_6, \dots, v_{11}\}$, and $\{v_{13}, v_{14}, v_{15}\}$ are quasi-levels but $\{v_5, \dots, v_{11}\}$ and $\{v_4, v_5, v_6\}$ are not. In the first counterexample, $v_{11} \in I$, but $p(v_{11}) = v_5$, the least element of I, and in the second counterexample, $p(v_7) < v_4 < v_7$ and v_4 is the least element of I, but $v_7 \notin I$.

The interpretation τ. We define a 2-dimensional interpretation $\tau : \Gamma_6 \to \Gamma_3$. Let

$$(x_1, y_1, z_1, x_2, y_2, z_2)$$

be the constants in Γ_6 and (x, y, z) be the constants in Γ_3. Given $G \in \mathcal{G}_3$, the domain of G^τ consists of "two copies" of G, which we achieve by $\partial^\tau(u, v) \iff v = x \vee v = y$. Within each copy, we inherit the edge relation from G, and we let, e.g., x_i be the vertex corresponding to x in copy i. We do not put any edges between the two copies except for connecting x_1 and x_2. Notice that τ is quantifier-free.

▶ **Definition 16.** *Let $\mathcal{G}'_6$ be $\{G^\tau : G \in \mathcal{G}'_3\}$.*

Then (by definition), τ is an interpretation $\mathcal{G}'_6 \to \mathcal{G}'_3$. Moreover,

▶ **Theorem 17.** *Let $\mathcal{B}$ be the set of all expansions of graphs in $\mathcal{G}'_6$ by a breadth-first traversal. There is a $(\mathcal{G}'_6, \mathcal{B})$-invariant formula ψ such that for any $(G, x, y, z) \in \mathcal{G}'_3$,*

$$d(x, y) = d(x, z) \implies G^\tau \models (\mathfrak{B} <)\, \psi$$

$$|d(x, y) - d(x, z)| \geq 2 \implies G^\tau \models \neg(\mathfrak{B} <)\, \psi,$$

where the second case also contains all those graphs where x, y, and z are not all connected.

(The proof is deferred to the Appendix.)

By composing the interpretation ρ with the interpretation τ, we see that for any successor expansion of a finite 2-pointed directed graph $D \in \mathcal{D}_2^S$, there is a path from s to t in D if and only if $D^{\rho\tau} \models (\mathfrak{B} <)\, \psi$. Hence,

▶ **Corollary 18.** *The directed reachability query is BFT-invariant definable over $\mathcal{D}_2^S$.*

3 Descriptive Complexity

In this section we obtain the main results of this paper: a characterization of deterministic and nondeterministic logarithmic space by traversal and breadth-first traversal invariance quantifiers respectively.

3.1 Multihead finite automata

A *nondeterministic multihead finite automaton* (NMFA) is an automaton with a single tape, finitely many heads on that tape, and a finite control. Unlike a Turing machine, the tape is not infinite; rather, it is initialized to the input string plus two special characters on either side to mark the left and right endpoints. Also unlike a Turing machine, the heads cannot write, they can only move left, right, or stay put depending on which characters they are reading. A single state is designated as accepting; if the computation enters this state then we say it halts. The language of an NMFA is exactly the set of strings it halts on.

Formally, an NMFA consists of a set Q of states, some number $k \in \mathbb{N}$ of heads, an input alphabet Σ, a start state $q_0 \in Q$, an accept state $q_f \in Q$, and a transition relation δ which relates k-tuples in $\Sigma \cup \{\triangleright, \triangleleft\}$ with $\{-1, 0, 1\}^k$. If any head is reading the left (respectively right) endpoint character, no subsequent transition may move that head right (respectively left). Futhermore, if the current state is q_f, then the transition relation moves all heads to the left (if possible) or fixes them if they are already at the left endpoint.

A *configuration* of an NMFA consists of the input string, the current state, and the location of the heads. The transition relation induces a relation on the space of configurations in the natural way. The *intial configuration* is the one in which the state is q_0 and all heads are at the left. The *final configuration* is the same but with state q_f. By the stipulation of the transition relation, if an NMFA enters q_f, then it will always enter the final configuration.

The *configuration graph* of an NMFA on a particular input x is a 2-pointed directed graph whose vertices are the set of configurations on x and whose edge relation is the graph of the relation induced by the transition function. The source and sink are the initial and final configurations respectively.

Strings and pointed graphs as structures. Let Γ_2 be the language of 2-pointed graphs, and let (Γ_2, S) be the language of 2-pointed successor graphs, with two (additonal) constants min and max, and a successor function.

Let Σ be a finite alphabet. We think of a string $x = x_0 x_1 \ldots x_{n-1}$ in $\Sigma^\star$ as a finite structure with domain $\{0, 1, \ldots, n-1\}$, a predicate σ for each $\sigma \in \Sigma$ with semantics

$$(\forall i < n)\ x \models \sigma(i) \iff x_i = \sigma,$$

constants min and max naming 0 and $n-1$, and finally a successor function taking index i to index $i+1$.[6]

We henceforth overload the meaning of Σ to indicate not only an alphabet, but also the signature of strings in that alphabet, so that the terms "finite Σ-structure" and "member of $\Sigma^\star$" denote the same objects.

Crucial to our work is that for a fixed NMFA, there is an interpretation that takes an input string and translates it into the associated configuration graph.

▶ **Theorem 19.** *For every NMFA $\mathcal{M}$ with alphabet Σ there is an interpretation $\pi : \Gamma_2 \to \Sigma$ such that for every sufficiently long string $x \in \Sigma^\star$, x^π is isomorphic to the configuration graph of $\mathcal{M}$ on input x.*

Furthermore, we can expand π to an interpretation $\pi : (\Gamma_2, S) \to \Sigma$, so that x^π is a successor expansion of the above configuration graph.

Moreover, π can be made quantifier-free.

(The proof is deferred to the Appendix)

▶ **Definition 20.** *An NMFA $\mathcal{M}$ is* symmetric *(SMFA) in case, for any input x, the configuration graph of $\mathcal{M}$ on x is undirected.*

[6] This is a common, "folklore," method of representing strings as structures. Often one takes a total ordering $<$ over the indices of a string instead of the successor function (see Libkin [7]), but for our purposes, either will work.

Computability by NMFAs is known to capture exactly nondeterministic logarithmic space (NL) [4], and computability by SMFAs captures at least logarithmic space (L).[7]

3.2 Capturing L and NL

Canonical encodings. For any finite structure A, any linear order $(A,<)$, and any fixed alphabet Σ of size at least 2, there is a *canonical encoding* of $(A,<)$ as a string in $\Sigma^\star$.

This construction is the foundation of all results in descriptive complexity, and can be found in numerous texts, e.g., [7]. We will not repeat it here. We do note, however, that any successor expansion (A,S) of A induces a linear order – hence every successor expansion of any finite structure has a canonical encoding.

Even more importantly, this canonical encoding is *definable* as the translation induced by a quantifier-free interpretation. The details are complicated, but can be found in Section 9.2 of [7].

▶ **Theorem 21.** *For every signature L, there is a quantifier-free interpretation $\mu : \Sigma \to (L,S)$ such that for every successor expansion (A,S) of any finite L-structure A, $(A,S)^\mu$ is the canonical encoding of (A,S).*

We now state the definition of a complexity-bounded query over finite structures, for which we need to imagine models of computation that take finite structures as input. We follow the standard method in descriptive complexity, which is to take a model of computation that operates on strings, and feed it the encoding $(A,S)^\mu$ of a structure A. Of course this encoding is not canonical given only A; for a well-defined query, we demand that the result of the computation is invariant of the particular expansion (A,S).

Now we are in a position to state:

▶ **Theorem 22.** *For every signature L and every logarithmic space query Q over finite L-structures, there is a quantifier-free interpretation $\gamma : \Gamma_2 \to (L,S)$ such that for every sufficiently large finite L-structure A,*

$$A \in Q \iff A \models (\mathfrak{S} S)\,((\mathfrak{T} <)\,\varphi)^\gamma,$$

where $(\mathfrak{T} <)\,\varphi$ is the sentence in the language of 2-pointed ordered graphs asserting that the distinguished vertices are connected.

Proof. Let $\mu : \Sigma \to (L,S)$ be the interpretation given by L in Theorem 21, let $\mathcal{M}$ be an SMFA deciding Q, let $\pi : \Gamma_2 \to \Sigma$ be the associated interpretation from Theorem 19, and let $\gamma = \mu\pi$. Fix a finite L-structure A and an arbitrary successor expansion (A,S).

Then $\mathcal{M}$ accepts the string $(A,S)^\mu$ iff $A \in Q$. But, $(A,S)^{\mu\pi} = (A,S)^\gamma$ is the configuration graph of $\mathcal{M}$ on $(A,S)^\mu$, so $\mathcal{M}$ accepts $(A,S)^\mu$ just in case the distinguished vertices of $(A,S)^\gamma$ are connected. In other words,

$$A \in Q \iff (A,S)^\gamma \models (\mathfrak{T} <)\,\varphi.$$

Therefore,

$$A \in Q \iff (A,S) \models ((\mathfrak{T} <)\,\varphi)^\gamma,$$

and since the right hand side is independent of the particular successor expansion,

$$A \in Q \iff A \models (\mathfrak{S} S)\,((\mathfrak{T} <)\,\varphi)^\gamma. \qquad ◀$$

[7] In [1], Axelsen considers the more restrictive *reversible* MFAs, which are both deterministic and backwards deterministic, and shows that they capture L. It is plausible that this might allow us to strengthen our results even further.

▶ **Theorem 23.** *For every signature L and every* NL *query Q over finite L-structures, there is a quantifier-free interpretation $\gamma : \Gamma_6 \to (L, S)$ such that for every sufficiently large finite L-structure A,*

$$A \in Q \iff A \models (\mathfrak{S}S)\,((\mathfrak{B} <)\,\psi)^{\gamma},$$

where ψ is the sentence from Theorem 17 in the language $(\Gamma_6, <)$.

Proof. Let $\mu : \Sigma \to (L, S)$ be the interpretation given by L in Theorem 21, let $\mathcal{M}$ be an NMFA deciding Q, let $\pi : (\Gamma_2, S) \to \Sigma$ be the associated interpretation from Theorem 19. Recall the interpretations $\rho : \Gamma_3 \to (\Gamma_2, S)$ and $\tau : \Gamma_6 \to \Gamma_3$ from Section 2.2. Finally, let $\gamma = \mu\pi\rho\tau$. Fix a finite L-structure A and an arbitrary successor expansion (A, S).

Then $\mathcal{M}$ accepts the string $(A, S)^{\mu}$ iff $A \in Q$. But $\mathcal{M}$ accepts $(A, S)^{\mu}$ just in case there is a path from source to sink over the graph $(A, S)^{\mu\pi}$. By Corollary 18, this occurs just in case $(A, S)^{\mu\pi\rho\tau} \models (\mathfrak{B} <)\,\psi$. But $(A, S)^{\mu\pi\rho\tau} = (A, S)^{\gamma}$.

Since the above is independent of the particular successor expansion S,

$$A \in Q \iff A \models (\mathfrak{S}S)((\mathfrak{B} <)\,\psi)^{\gamma},$$

which completes the proof. ◀

3.3 Logspace-computable traversals

In the other direction, we want to show that traversals and breadth-first traversals are computable in L and NL respectively. These constructions rely on the computability in logarithmic space of undirected *st*-connectivity, and furthermore on the existence of logarithmic space *universal exploration sequences* [8].

Given an ordered graph G and a vertex v, it is possible to construct, in logarithmic space, the index of the least vertex u in the connected component of v. Simply iterate through the vertices of G in order, testing connectivity with v, until we find a vertex that is connected.

▶ **Theorem 24.** *There is a logarithmic space Turing machine which, for every finite ordered graph $(G, <)$, computes a traversal $(G, \prec)$ in the following sense: given the canonical encoding of $(G, <)$ and (indices of) two of its vertices v and w, accepts or rejects according to whether $v \prec w$.*

Proof. Given two vertices v and w in G, first test whether they are in the same connected component. If not, let v_0 and w_0 be the least elements in the connected components of v and w respectively, and compare v and w according to whether $v_0 < w_0$.

Otherwise, let v_0 be the least element of their common connected component. If $n = |G|$, construct (using space logarithmic in n) a universal exploration sequence, and explore the connected component of v and w according to that sequence starting with v_0. Let $v \prec w$ iff the first occurrence of v precedes the first occurrence of w.

We must show $(G, \prec)$ is a traversal. Notice that connected components induce intervals. If v is not the least vertex in some connected component, then its first occurrence in the universal exploration sequence has some immediate predecessor u which is a neighbor. Therefore, in the traversal, $u \prec v$; hence, v has some preceding neighbor. ◀

Canonical BFT of an ordered graph. Unlike the case of ordinary traversals, where the traversal $(G, \prec)$ of $(G, <)$ depends on some family of universal exploration sequences, we will define a canonical breadth-first traversal $(G, \prec_B)$ of an ordered graph $(G, <)$ and show that it can be computed in nondeterminstic logspace.

▶ **Definition 25.** *Given a finite ordered graph $(G, <)$ and vertices $v, w \in G$, let v_0 and w_0 be the $<$-least elements of the connected components of v and w respectively. Let $<^\star$ be the ordering on finite sequences of vertices that orders them* first *by length, and* then *lexicographically. Let $\vec{v}$ be the $<^\star$-least path from v_0 to v. Then,*

1. *if $v_0 \neq w_0$, then $v \prec_B w \iff v_0 < w_0$,*
2. *if $v_0 = w_0$ then $v \prec_B w \iff \vec{v} <^\star \vec{w}$.*

▶ **Lemma 26.** *For any finite ordered graph $(G, <)$, $(G, \prec_B)$ is a breadth-first traversal.*

(Proof deferred to appendix.)

▶ **Theorem 27.** *There is a logarithmic space nondeterministic Turing machine which, on input a finite ordered graph $(G, <)$ and vertices $v, w \in G$, decides whether or not $v \prec_B w$.*

Proof. As in the proof of Theorem 24, given two vertices v and w, first test whether $v_0 = w_0$ (i.e., whether they're in the same connected component). If not, decide $v \prec_B w$ according to whether $v_0 < w_0$.

Otherwise we argue that we can construct the sequence $\vec{v} = (v_0, \dots, v_{\ell-1}, v)$ in the following sense: given an index for v_i, we can test whether it's equal to v; if not, we can construct the index of v_{i+1}, all in logarithmic space.

If we can do this, then we decide $v \prec_B w$ by comparing $\vec{v} <^\star \vec{w}$. First we compare their lengths: we simultaneously construct (v_{i+1}, w_{i+1}) from (v_i, w_i), until the first index is v or the second is w. Unless this happens at the same stage, we are done. (Since (v_{i+1}, w_{i+1}) overwrites (v_i, w_i), this remains in logarithmic space.)

Otherwise, we start over, and simultaneously construct (v_i, w_i) until we (necessarily) find the first index at which they differ. Then we decide $v \prec_B w$ according to which is larger.

It remains to show how to construct v_{i+1} from v_i. Orient all edges in G so that they increase distance from v_0. Then v_{i+1} is the $<$-least vertex x such that there is an edge (v_i, x) and a directed path (x, v). Since we can compute directed reachability in nondeterminstic logarithmic space, we can find v_{i+1} in nondeterminstic logarithmic space as well. ◀

At this point we are ready to state two of the central results of this paper.

▶ **Theorem 28.** *The following are equivalent:*

1. *Q is a logspace-decidable query over finite K-structures.*
2. *There is a quantifier-free interpretation $\pi : \Gamma_2 \to (K, S)$ such that for all sufficiently large finite K-structures A,*

$$A \in Q \iff A \models (\mathfrak{S} S)\, ((\mathfrak{T} <)\, \varphi)^\pi,$$

where φ is the formula expressing undirected st-connectivity.

3. *There is a traversal-invariant definable query R over finite (K, S) structures such that for any finite K-structure A,*

$$A \in Q \iff A \models (\mathfrak{S} S)\, R$$

Proof. Implication $1 \Rightarrow 2$ is exactly Theorem 22. Implication $2 \Rightarrow 3$ is immediate, as $((\mathfrak{T} <)\, \varphi)^\pi$ is by definition a traversal-invariant definable query, and traversal-invariant queries are closed under finite differences. It remains to show $3 \Rightarrow 1$.

It suffices to show that the traversal-invariant definable query R is logspace computable over finite (K, S) structures, as given an encoding of a K-structure A, a logspace Turing machine can always compute a successor relation on the domain of A, by using the particular encoding in which A is presented.

Since logspace-computable queries are closed under boolean combinations, it suffices to show that any basic traversal-invariant definable query is logspace computable. Since logspace computable queries are closed under elementary interpretations, it suffices to show that for any class $\mathcal{G}'$ of finite graphs, every $(\mathcal{G}', \mathcal{T}')$-invariant formula is logspace computable over graphs in $\mathcal{G}'$, where $\mathcal{T}'$ is the family of all expansions of graphs in $\mathcal{G}'$ by traversals.

But for this, it suffices to show that any first-order sentence in the language of ordered graphs is logspace computable given an encoding of a finite graph, where the order is the traversal defined in Theorem 24. Since logspace queries are closed under first-order combinations, it suffices to check that given any encoding of a graph and two vertices therein, we can test whether they are equal, test whether they are connected by an edge, or compare them according to the canonical traversal.

The first two are true, and the last is exactly Theorem 24. ◀

By replacing "L" by "NL" and "traversal" by "breadth-first traversal" throughout, we get

▶ **Theorem 29.** *The following are equivalent:*

1. *Q is an nlogspace-decidable query over finite K-structures.*
2. *There is a quantifier-free interpretation $\pi : \Gamma_6 \to (K, S)$ such that for all sufficiently large finite K-structures A,*

$$A \in Q \iff A \models (\mathfrak{S}S)\,((\mathfrak{B} <)\,\psi)^\gamma,$$

where ψ is the sentence of Theorem 17.

3. *There is a breadth-first traversal-invariant definable query R over finite (K, S) structures such that for any finite K-structure A,*

$$A \in Q \iff A \models (\mathfrak{S}S)\,R$$

Structures with successor. Suppose the signature K contains the unary function symbol S, and that $\mathcal{K}$ is a family of finite K-structures in which S is interpreted by a successor function. Then the quantifier $(\mathfrak{S}S)$ in any successor-invariant query $(\mathfrak{S}S)\,R$ is superfluous over $\mathcal{K}$; i.e., for any $A \in \mathcal{K}$,

$$A \models R \iff A \models (\mathfrak{S}S)\,R.$$

The reason is that the interpretation of S in R is independent of the particular successor function on A that we choose, so we might as well choose the one native to A.

In particular, for such families $\mathcal{K}$, we can drop the $(\mathfrak{S}S)$ quantifier from the traversal- or breadth-first traversal-invariant queries R of Theorems 28 and 29. In particular, let us take the case of strings over a finite alphabet Σ, which are the original setting for logspace and nlogspace queries, and also successor structures as described in Section 3. Then we have

▶ **Corollary 30.** *For any family $Q \subseteq \Sigma^\star$,*

1. *Q is logspace-decidable iff there is a traversal-invariant definable query R such that for every string $x \in \Sigma^\star$, $x \in Q \iff x \models R$, and*
2. *Q is nlogspace-decidable iff there is a breadth-first traversal-invariant definable query R such that for every string $x \in \Sigma^\star$, $x \in Q \iff x \models R$.*

3.4 Discussion and open questions

Our results are the first presentation-invariant characterizations of L and NL, and, to our knowledge, the largest known complexity classes characterized by first-order logic extended by invariant definability of an elementary class of presentations. They demonstrate the surprising power of interpretations (even quantifier-free ones!) and establish a new foundational correspondence between graph traversals and complexity classes.

The elephant in the room is whether *depth-first traversal invariance* captures a meaningful complexity class, like polynomial time. While we have been able to find depth-first invariant definitions of certain suggestive queries (like *vertex-avoiding paths*), we still do not have very strong evidence one way or the other. More generally, there are a variety of graph traversals and a variety of associated presentations (such as the ancestral relation of the traversal tree) which might correspond to interesting complexity classes.

Finally, we have extended these notions of definability to arbitrary infinite structures by requiring that the underlying order be well-founded. (Since well-orders are not elementarily definable, this circumvents the usual "Beth definability" obstacle to studying presentation invariance over infinite structures.) Whereas separating traversal-invariant from BFT-invariant definability over classes of finite structures requires separating L and NL, it is plausibly easier to separate them over arbitrary classes, and it is plausible that this will inform the finite case. This work is ongoing.

References

1 Holger Bock Axelsen. Reversible multi-head finite automata characterize reversible logarithmic space. In Adrian-Horia Dediu and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications*, pages 95–105, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

2 Derek Corneil and Richard Krueger. A unified view of graph searching. *SIAM J. Discret. Math.*, 22(4):1259–1276, July 2008.

3 Yuri Gurevich. Toward logic tailored for computational complexity. In M. M. Ricther et al., editor, *Computation and Proof Theory, Lecture Notes in Mathematics 1104*, pages 175–216. Springer-Verlag, 1984.

4 Juris Hartmanis. On non-determinancy in simple computing devices. *Acta Informatica*, 1(4):336–344, December 1972.

5 Wilfrid Hodges. *Model Theory.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1993.

6 Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

7 Leonid Libkin. *Elements of Finite Model Theory.* Springer, Berlin, Heidelberg, 2004.

8 Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008.

9 Nicole Schweikardt. A short tutorial on order-invariant first-order logic. In *CSR 2013*, June 2013.

10 Martin Tompa. *Introduction to Computational Complexity.* Lecture notes, 1991. URL: `https://homes.cs.washington.edu/~tompa/papers/532.pdf`.

11 Moshe Vardi. The complexity of relational query languages. In *STOC82*, pages 137–146, 1982.

A Interpretations and change of signature

We review the basic definitions behind interpretations, following the exposition of Hodges [5], except that we also allow for functional signatures (see below). There is no new mathematical content here; however, getting the definitions and terminology straight is terribly important, since we use interpretations extensively.

▶ **Definition 31.** *Let L and K be signatures and $k \in \mathbb{N}$. An* elementary k-ary interpretation *$\pi : L \to K$ is a first-order K-formula $\partial^\pi(\bar{x})$, for each constant symbol $c \in L$ a variable-free K-term c^π, and for each relation symbol $r \in L$, a first-order K-formula $r^\pi(\bar{x}_1, \ldots, \bar{x}_n)$, where n is the arity of r, and the length of each tuple throughout is k.*

An interpretation is quantifier-free *in case ∂^π and each r^π is quantifier-free.*

▶ **Definition 32.** *Given an elementary k-ary interpretation $\pi : L \to K$ and a first-order L-term or L-formula α, its* translation *α^π, is a K-formula given by the following recursion:*

1. *If α is a variable x, then α^π is a k-tuple of (distinct) variables $\bar{x}$.*
2. *If α is a constant symbol c, then α^π is c^π.*
3. *If α is the atomic formula $r(\alpha_1, \ldots, \alpha_n)$, then α^π is $r^\pi(\alpha^\pi, \ldots, \alpha^\pi)$,*
4. *If α is a boolean combination of formulas ϑ, then α^π is the same boolean combination of formulas ϑ^π,*
5. *If α is $\exists x\, \vartheta$, then α^π is the formula $\exists \bar{x}\, \partial^\pi(\bar{x}) \wedge \vartheta^\pi$, and*
6. *If α is $\forall x\, \vartheta$, then α^π is the formula $\forall \bar{x}\, \partial^\pi(\bar{x}) \to \vartheta^\pi$.*

In the definition below, $\partial^\pi[A^k]$ denotes the subset of A^k on which ∂^π holds.

▶ **Definition 33.** *Suppose $\pi : L \to K$ is an interpretation and A is a K-structure. Then the* π-translation *A^π is the L-structure with domain $\partial^\pi[A^k]$ with the denotation of λ given by λ^π, for each $\lambda \in L$.*

(Note that even though the arity of λ^π is nk as a K-formula, it defines an n-ary relation over A^π, whose elements are k-tuples of A.)

Functional signatures. In a very particular case (see *successor expansions*, Definition 10) we will want to consider signatures with function symbols, and exactly once (Theorem 19), we will want to define an interpretation $\pi : L \to K$ where L has some function symbol $f(x_1, \ldots, x_n)$. In this case $f^\pi(\vec{x}_1, \ldots, \vec{x}_n)$ is a *definition by cases*, where each case is a first-order K-formula, and the *definiens* inside each case is a k-tuple of K-terms in the free variables $(\vec{x}_1, \ldots, \vec{x}_n)$, where k is the arity of π. In a quantifier-free interpretation, each case must be a quantifier-free K-formula.

Functional signatures also generalize signatures with constants, which are nullary function symbols. For a constant symbol c, c^π is a definition by cases, where each case is a k-tuple of variable-free K-terms.

It is common practice in finite model theory to replace functions by their graph relations, thus working with purely relational signatures. The only reason for considering functional signatures here is to make certain interpretations quantifier free (cf. Theorems 28 and 29); in the purely relational setting, these interpretations would contain quantifiers.

Injective interpretations. Usually an interpretation will also contain a first-order K-formula $eq^\pi(\bar{x}, \bar{y})$ defining when we regard two k-tuples as equal. (For example, when interpreting rational numbers by pairs of integers, we say $(a, b) = (c, d) \iff ac - bd = 0$.) In case eq^π is simply equality of tuples (as above), π is called an *injective interpretation.* Here we do not deal with any interpretations with a nontrivial equivalence relation. Therefore, it is convenient to drop the word “injective” and simply refer to interpretations.

▶ **Lemma 34** (Fundamental property of interpretations)**.** *Suppose that $\pi : L \to K$ is an elementary interpretation. Then for every K-structure A, every n-ary L-sentence φ, and every $\bar{x}_1, \ldots, \bar{x}_n$ in the domain $\partial^\pi[A^k]$ of A^π,*

$$A \models \varphi^\pi(\bar{x}_1, \ldots, \bar{x}_n) \iff A^\pi \models \varphi(\bar{x}_1, \ldots, \bar{x}_n).$$

Note that on the left-hand side, $(\bar{x}_1, \ldots, \bar{x}_n)$ is regarded as an nk-tuple of elements in A, and on the right-hand side, it is regarded as an n-tuple of elements in A^π.

▶ **Definition 35.** *Suppose that $\pi : L \to K$, $\mathcal{L}$ is a class of L-structures, and $\mathcal{K}$ is a class of K-structures. Then π is an* interpretation $\mathcal{K} \to \mathcal{L}$ *in case for every $A \in \mathcal{K}$, $A^\pi \in \mathcal{L}$.*

Properties of interpretations ρ and τ

Lemma 13. For any graph $D \in \mathcal{D}_2^S$, there is a directed path from s to t in D iff the vertices y and z are equidistant from x in D^ρ. Even stronger, if there is no directed path from s to t in D, then either $d(x, z)$ is undefined or $|d(x, y) - d(x, z)| \geq 2$, where d indicates distance in D^ρ.

Proof of Lemma 13. (Adapted from [10]) Fix a graph D and let n be the number of vertices in D. Identify the vertices of D with $\{0, 1, \ldots, n-1\}$ such that $S(i, i+1)$. Then in D^ρ, there is a path

$$x = (s, 0) - (s, 1) - \cdots - (s, n-1) = y,$$

of length $n-1$, and this is moreover the distance between x and y, by considering the second coordinate.

If t is reachable from s in D, then that must be witnessed by some directed path $(s = r_0 \to r_1 \to \cdots \to r_{\ell-1} = t)$ of length $\ell \leq n$. Then

$$(r_0, 0) - (r_1, 1) - \cdots - (r_{\ell-1}, \ell - 1) - (r_{\ell-1}, \ell) - \cdots - (r_{\ell-1}, n-1)$$

is a path in D^ρ from x to z of length exactly $n-1$. Again by considering the second coordinate, we can see that there is no shorter path. Hence y and z are equidistant from x.

Conversely, suppose that there were a path in D^ρ from x to z in D^ρ of length exactly $n-1$. Then it must be of the form

$$(u_0, 0) - (u_1, 1) - \cdots - (u_{n-1}, n-1),$$

where $u_0 = s$, $u_{n-1} = t$, and for each i, either $u_i = u_{i+1}$ or $u_i \to u_{i+1}$ in D. Hence the u_i witness a directed path from s to t.

Moreover, observe the parity of the second coordinate in any path must alternate. Hence, the length of any path from $(s, 0)$ to $(t, n-1)$ must be equal to $n-1$ modulo 2. Therefore, if there is no directed path from s to t in D, then in D^ρ then any path from x to z in D^ρ must have length at least $n+1$. This is at least 2 greater than $d(x, y)$, which is $n-1$. ◀

Theorem 17. Let $\mathcal{B}$ be the set of all expansions of graphs in $\mathcal{G}_6'$ by a breadth-first traversal. There is a $(\mathcal{G}_6', \mathcal{B})$-invariant formula ψ such that for any $(G, x, y, z) \in \mathcal{G}_3'$,

$$d(x, y) = d(x, z) \implies G^\tau \models (\mathfrak{B} <)\, \psi$$

$$|d(x, y) - d(x, z)| \geq 2 \implies G^\tau \models \neg(\mathfrak{B} <)\, \psi,$$

where the second case also contains all those graphs where x, y, and z are not all connected.

Proof of Theorem 17. Let ψ assert that all six constants $(x_1, \ldots, z_2)$ occur in the same connected component; moreover, if $x_1 < x_2$, then y_2 and z_2 occur in the same quasi-level, and if $x_2 < x_1$, then y_1 and z_1 occur in the same quasi-level. (To show that ψ is invariant, it suffices to show that ψ is correct.)

Fix a graph $(G, x, y, z) \in \mathcal{G}'_3$ such that $d(x, y) = d(x, z)$. Consider its translation, and expand this by an arbitrary breadth-first traversal. We may assume that all constants $(x_1, \dots, z_2)$ lie in the same connected component; otherwise ψ evaluates to false, which is correct as not all of (x, y, z) are connected.

Suppose that $x_1 < x_2$. Let w be the least element of $<$ in the connected component of x_1. Then w must be in G_1, so any path from w to y_2 or z_2 must pass through the edge (x_1, x_2). Hence,

$$|d(x, y) - d(x, z)| = |d(x_2, y_2) - d(x_2, z_2)| = |d(w, y_2) - d(w, z_2)|.$$

In other words, the desired quantity is exactly the difference in distance between y_2 and x_2 to the source. We know that this difference is either equal to 0 or at least 2, and ψ correctly distinguishes these two cases by testing whether y_2 and z_2 occur in the same quasi-level.

Similarly, if $x_2 < x_1$, ψ distinguishes $|d(x, y) - d(x, z)| = 0$ from $|d(x, y) - d(x, z)| \geq 2$ by testing whether y_1 and z_1 occur in the same quasi-level. ◀

B Defining configuration graphs by interpretation

Theorem 19. For every NMFA $\mathcal{M}$ with alphabet Σ there is an interpretation $\pi : \Gamma_2 \to \Sigma$ such that for every sufficiently long string $x \in \Sigma^\star$, x^π is isomorphic to the configuration graph of $\mathcal{M}$ on input x. Furthermore, we can expand π to an interpretation $\pi : (\Gamma_2, S) \to \Sigma$, so that x^π is a successor expansion of the above configuration graph. Moreover, π can be made quantifier-free.

Proof of Theorem 19. Let k be the number of heads in $\mathcal{M}$. Then $k+1$ will be the dimension of π. The domain of the interpretation ∂_π just stipulates that the first coordinate is less than q, the number of states of $\mathcal{M}$.

We can now establish a bijection between the domain of π and configurations of $\mathcal{M}$ with input x, for any string x such that $|x| \geq q$. A configuration is simply specified by current state and the location of the heads, which correspond to the first and remaining k coordinates of the domain respectively. Since x is sufficiently long, there are enough choices in the first coordinate for all states of $\mathcal{M}$.

Let $\vec{u}$ and $\vec{v}$ be arbitrary configurations of $\mathcal{M}$ on input an arbitrary string of length at least q. We want to define $E^\pi(\vec{u}, \vec{v})$ to hold just in case the configuration $\vec{v}$ is reachable from $\vec{u}$ in one step. This is definable by a boolean combination of formulas of the following form:

1. u_i is the minimum or maximum index,
2. $\sigma \in \Sigma$ is the character at index, and
3. indices u_i and v_i are identical or adjacent.

Each of these formulas is quantifier-free definable in the language Σ, by e.g.,

1. $u_i = 0$ or $u_i = n - 1$,
2. $\sigma(u_i)$, and
3. $u_i = v_i$ or $S(u_i) = v_i$ or $u_i = S(v_i)$

respectively, where 0 and $n - 1$ are aliases for min and max respectively. Finally,

$$s^\pi = (0, 0, \dots, 0), \; t^\pi = (S(0), 0, \dots, 0).$$

This is because all heads are at the left in the initial or final configuration, 0 is the start state, and 1 is the halt state.

Now for any string x of length at least q, not only is the domain of x^π in bijection with the configurations of $\mathcal{M}$ on x, but relative to this bijection π_E defines the graph of "next," and π_s and π_t are the initial and final configurations respectively. Hence x^π as a structure is isomorphic to the configuration graph of $\mathcal{M}$ on input x.

To expand π to an interpretation from Γ_2^S, we need to define a successor function on $(k+1)$-tuples of indices, given a successor function on indices. This is easy to do by mimicking the standard "increment-by-one" algorithm on numbers written in some fixed radix.[8] ◀

First-order definitions of traversals

Lemma 26. For any finite ordered graph $(G,<)$, $(G,\prec_B)$ is a breadth-first traversal.

Proof of Lemma 26. Connected components of G induce intervals of $(G,\prec_B)$, so it suffices to assume that G is connected. Let v_0 be the least element of G (unambiguously with respect to either order).

Suppose v is a non-minimal vertex and let $\vec{v} = (v_0, v_1, \dots, v_{\ell-1}, v)$. Since $<^\star$-least shortest paths are closed under prefixes, $v_i \prec_B v$ for each v; in particular, $|\vec{v}_{\ell-1}| = \ell - 1$.

Let u be the $\prec_B$-least neighbor of v, and $\vec{u}$. Since $u \preceq_B v_{\ell-1}$, $|\vec{u}| \leq \ell - 1$. Since $(\vec{u}, v)$ is a path from v_0 to v of length at most ℓ, and since ℓ is the distance from v_0 to v, $|\vec{u}| = \ell - 1$.

Since u and $v_{\ell-1}$ are the same distance from v_0, we have

$$\vec{u} \leq^\star (v_0, \dots, v_{\ell-1}) \wedge (v_0, \dots, v_{\ell-1}, v) \leq^\star (\vec{u}, v).$$

Therefore $\vec{u} = (v_0, \dots, v_{\ell-1})$. In particular, $u = v_{\ell-1}$.

Finally, suppose that v and w are arbitrary non-minimal vertices of G, and that $v \prec_B w$. Let $v_\dagger$ and $w_\dagger$ be the second-to-last elements of $\vec{v}$ and $\vec{w}$ respectively. Then $v_\dagger$ and $w_\dagger$ are the $\prec_B$-least neighbors of v and w, so it suffices to show that $v_\dagger \preceq_B w_\dagger$.

However, $\vec{v} = (\vec{v}_\dagger, v)$ and $\vec{w} = (\vec{w}_\dagger, w)$. Since $\vec{v} <^\star \vec{w}$, $\vec{v}_\dagger \leq^\star \vec{w}_\dagger$, which concludes the proof. ◀

[8] This is the only point in which we have to define S^π where S is a *function symbol*, here we use a definition by cases in which every case is a quantifier-free term guarded by a quantifier-free formula.

Semiring Provenance in the Infinite

Sophie Brinke
RWTH Aachen University, Germany

Erich Grädel
RWTH Aachen University, Germany

Lovro Mrkonjić
RWTH Aachen University, Germany

Matthias Naaf
RWTH Aachen University, Germany

Abstract

Semiring provenance evaluates database queries or logical statements not just by true or false but by values in some commutative semiring. This permits to track which combinations of atomic facts are responsible for the truth of a statement, and to derive further information, for instance concerning costs, confidence scores, number of proof trees, or access levels to protected data. The focus of this approach, proposed and developed to a large extent by Val Tannen and his collaborators, has first been on (positive) database query languages, but has later been extended, again in collaboration with Val, to a systematic semiring semantics for first-order logic (and other logical systems), as well as to a method for the strategy analysis of games.

So far, semiring provenance has been studied for finite structures. To extend the semiring provenance approach for first-order logic to infinite domains, the semirings need to be equipped with addition and multiplication operators over infinite collections of values. This needs solid algebraic foundations, and we study here the necessary and desirable properties of semirings with infinitary operations to provide a well-defined and informative provenance analysis over infinite domains. We show that, with suitable definitions for such infinitary semiring, large parts of the theory of semiring provenance can be succesfully generalised to infinite structures.

2012 ACM Subject Classification Theory of computation → Finite Model Theory

Keywords and phrases Semiring semantics, first-order logic, semirings with infinitary operations

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.3

1 Introduction

Semiring provenance was proposed in 2007 by Val Tannen in the seminal paper [7], together with Todd Green and Grigoris Karvounarakis. It is based on the idea to annotate the atomic facts in a database by values in some commutative semiring, and to propagate these values through a database query, keeping track whether information is used alternatively or jointly. This approach has been successfully applied to many variants of database queries, including conjunctive queries, positive relational algebra, datalog, nested relations, XML, SQL-aggregates, graph databases (see, e.g., the surveys [8, 3]). Depending on the chosen semiring, provenance valuations give practical information about a query, beyond its truth or falsity, for instance concerning the *confidence* that we may have in its truth, the *cost* of its evaluation, the number of successful evaluation strategies, and so on. Beyond such provenance evaluations in specific *application semirings*, more precise information is obtained by evaluations in *provenance semirings* of polynomials or formal power series, which permit us to *track* which atomic facts are used (and how often) to compute the answer to the query.

While semiring provenance had for a long time been restricted to negation-free query languages, a new approach for dealing with negation has been proposed in 2017 by Grädel and Tannen [4], based on transformations into negation normal form, quotient semirings of polynomials with dual indeterminates, and a close relationship to semiring valuations of games. In particular, this provides a semiring provenance analysis for full first-order logic (over finite domains). Further such a provenance analysis can be applied to many other logics and query languages with negation, and also permits a reverse provenance analysis, i.e., finding models that satisfy various properties under given provenance tracking assumptions, with potential applications to explaining missing query answers or failures of integrity constraints, and to using these explanations for computing repairs. An updated exposition of this approach can be found in [6].

If we investigate semiring provenance, beyond applications to finite data, as a general semiring-based semantics for first-order logic (and other logical systems), the question arises whether this semantics also makes sense over infinite domains, and what properties of the underlying semirings are needed to make such an extension possible and meaningful. This is the question that we want to study in this paper.

The obvious problem is the interpretation of quantifiers. A semiring interpretation π, over the universe A, assigns to a formula $\psi(\bar{a})$ a value $\pi[\![\psi(\bar{a})]\!]$ in some commutative semiring $\mathcal{S}$. This is defined by induction on ψ, and for the quantifiers, we have that

$$\pi[\![\exists x\varphi(x,\bar{b})]\!] := \sum_{a\in A} \pi[\![\varphi(a,\bar{b})]\!] \qquad \text{and} \qquad \pi[\![\forall x\varphi(x,\bar{b})]\!] := \prod_{a\in A} \pi[\![\varphi(a,\bar{b})]\!],$$

so for infinite universes, we need to equip the semirings with infinitary addition and multiplication operations, with suitable algebraic properties.

In some cases this is completely straightforward and unproblematic, for instance for finite min-max semirings or, more generally, for semirings induced by some complete lattice (with suprema and infima as semiring operations). There are other semirings, for instance the natural semiring $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$, which do not admit infinitary operations, but which can be easily completed to one that does so, such as $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$ where there is an obvious natural definition for infinitary addition and multiplication. But such extensions are not always obvious, for instance for semirings of polynomials. Further there are important semirings, such as the tropical semiring $\mathbb{T} = (\mathbb{R}_+^\infty, \min, +, \infty, 0)$ where the definition of the infinitary operations (here infimum and infinitary sum) is obvious, but it is not clear whether all relevant algebraic properties of the semiring operations also hold for their infinitary versions. In the case of the tropical semiring, we shall see that most of the basic algebraic properties do generalise to the infinite, with the exception of the distributive law which, in its strong form, does only hold on countable domains, but not on uncountable ones. Of course this poses the questions, what algebraic properties of infinitary semiring operations are actually needed for a well-defined and meaningful semiring semantics. We shall systematically study necessary and desirable algebraic properties of such infinitary operations and, on this basis, propose a definition of infinitary semirings. We will discuss examples of such semirings, focussing on one side on the case of absorptive infinitary semirings and, on the other side, define extensions of the polynomial semiring $\mathbb{N}[X]$ to a semiring of generalised power series, for which we can establish a universality property, similar to the one of $\mathbb{N}[X]$ in the finite case.

Using these infinitary semirings, we shall discuss semiring provenance for first-order logic on possibly infinite structures and show that a large part of the theory developed for the finite case does indeed carry over to infinite domains. In particular, we establish that the Sum-of-Proof-Trees-Theorem, saying that the semiring valuation of a first-order sentence coincides with the sum of the valuations of its proof trees also holds on infinite domains, provided that the underlying infinitary semiring satisfies an appropriate distributivity property.

2 Commutative Semirings

▶ **Definition 1** (Semiring)**.** *A commutative semiring is an algebraic structure* $\mathcal{S} = (S, +, \cdot, 0, 1)$ *with* $0 \neq 1$*, such that* $(S, +, 0)$ *and* $(S, \cdot, 1)$ *are commutative monoids,* $\cdot$ *distributes over* $+$*, and* $0 \cdot s = s \cdot 0 = 0$*.*

In this paper, we only consider commutative semirings and simply refer to them as *semirings*. A semiring is *naturally ordered* (by addition) if $s \leq t :\Leftrightarrow \exists r(s + r = t)$ defines a partial order. Notice that $\leq$ is always reflexive and transitive, so a semiring is naturally ordered if, and only if $\leq$ is antisymmetric, i.e. $r \leq s$ and $s \leq r$ only hold for $s = r$. In particular, this excludes rings.

A semiring $\mathcal{S}$ is *idempotent* if $s + s = s$ for each $s \in S$ and *multiplicatively idempotent* if $s \cdot s = s$ for all $s \in S$. If both properties are satisfied, we say that $\mathcal{S}$ is fully idempotent. Finally, $\mathcal{S}$ is *absorptive* if $s + st = s$ for all $s, t \in S$ or, equivalently, if multiplication is decreasing in $\mathcal{S}$, i.e. $st \leq s$ for all $s, t \in S$. Every absorptive semiring is idempotent, and every idempotent semiring is naturally ordered.

Application semirings. There are many applications which can be modelled by semirings and provide useful practical information about the evaluation of a formula.

- The Boolean semiring $\mathbb{B} = (\mathbb{B}, \vee, \wedge, \bot, \top)$ is the standard habitat of logical truth.
- A totally ordered set $(S, \leq)$ with least element s and greatest element t induces the *min-max semiring* $(S, \max, \min, s, t)$. Specific important examples are the Boolean semiring, the fuzzy semiring $\mathbb{F} = ([0, 1], \max, \min, 0, 1)$, and the *access control semiring*, also called the *security semiring* [2].
- A more general class (than min-max semirings) is the class of *lattice semirings* $(S, \sqcup, \sqcap, s, t)$ induced by a bounded distributive lattice $(S, \leq)$. Clearly, lattice semirings are fully idempotent.
- The *tropical semiring* $\mathbb{T} = (\mathbb{R}_{+}^{\infty}, \min, +, \infty, 0)$ is used to annotate atomic facts with a cost for accessing them and to compute minimal costs for verifying a logical statement. It is not fully idempotent but absorptive.
- The *Viterbi semiring* $\mathbb{V} = ([0, 1]_{\mathbb{R}}, \max, \cdot, 0, 1)$, which is in fact isomorphic to $\mathbb{T}$ via $y \mapsto -\ln y$ can be used for reasoning about confidence.
- An alternative semiring for this is the *Łukasiewicz semiring* $\mathbb{L} = ([0, 1]_{\mathbb{R}}, \max, \odot, 0, 1)$, where multiplication is given by $s \odot t = \max(s + t - 1, 0)$. It is isomorphic to the semiring of doubt $\mathbb{D} = ([0, 1]_{\mathbb{R}}, \min, \oplus, 1, 0)$ with $s \oplus t = \min(s + t, 1)$. Both $\mathbb{L}$ and $\mathbb{D}$ are absorptive semirings.
- The *natural semiring* $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$ is used to count the number of proof trees or evaluation strategies that estabish the truth of a sentence. It is also important for bag semantics in databases.

Provenance semirings. Provenance semirings of polynomials provide information on which combinations of literals imply the truth of a formula. The universal provenance semiring over a finite set X is the semiring $\mathbb{N}[X]$ of multivariate polynomials with indeterminates from X and coefficients from $\mathbb{N}$. Other provenance semirings are obtained, for example, as quotient semirings of $\mathbb{N}[X]$ induced by congruences for idempotence and absorption. The resulting provenance values are less informative but their computation is more efficient.

- By dropping coefficients from $\mathbb{N}[X]$, we get the free idempotent semiring $\mathbb{B}[X]$ whose elements are (in one-to-one correspondence with) finite sets of monomials with coefficient 1. It is the quotient induced by $x + x \sim x$.

- If, in addition, exponents are dropped, we obtain the Why-semiring $\mathbb{W}(X)$ of finite sums of monomials with coefficient 1 that are linear in each indeterminate. In this semiring, addition is idempotent but multiplication is not.
- The free absorptive semiring $\mathbb{S}(X)$ consists of $0, 1$ and all antichains of monomials with respect to the absorption order $\succcurlyeq$. A monomial m_1 absorbs m_2, denoted $m_1 \succcurlyeq m_2$, if it has smaller exponents, i.e. $m_2 = m \cdot m_1$ for some monomial m. It is the quotient of $\mathbb{N}[X]$ induced by $x + xy \sim x$.
- Finally, $(\text{PosBool}(X), \vee, \wedge, \bot, \top)$ is the semiring whose elements are classes of equivalent (in the usual sense) positive Boolean expressions with Boolean variables from X. Its elements are in bijection with the positive Boolean expressions in irredundant disjunctive normal form. This is the lattice semiring freely generated by the set X. It arises from $\mathbb{S}(X)$ by dropping exponents.

For treating logical formalisms with fixed-point constructions, such as Datalog or LFP, provenance semirings with more general objects than polynomials are needed (see [1, 7]. Examples include the semirings of formal power series (with possibly infinite sums of monomials) such as $\mathbb{N}^\infty[\![X]\!]$ and the semirings $\mathbb{S}^\infty(X)$ of generalised absorptive polynomials (admitting infinite exponents). Further, all these provenance semirings can be equipped with *dual indeterminates* for treating negation, see [6].

3 Semirings with Infinitary Operations

3.1 Basic Properties of an Infinitary Operation

We first treat the two semiring operations, addition and multiplication, separately, and then look at their connections. The properties of the individual operations are discussed in terms of addition, but apply to multiplication analogously.

So let $\mathcal{S} = (S, +, 0)$ be a commutative monoid which we want to expand by an infinitary operation $\sum$ that maps every sequence $(s_i)_{i \in I}$ (over an arbitrary index set I) to a value $\sum_{i \in I} s_i \in S$. The infinitary sum should be compatible with the finite sum and respect the basic algebraic properties of the monoid. We thus have the following requirements.

Partition invariance (infinite associativity): For each partition $(I_j)_{j \in J}$ of I we have

$$\sum_{i \in I} s_i = \sum_{j \in J} \sum_{i \in I_j} s_i.$$

Bijection invariance (infinite commutativity): For every bijection $\sigma\colon J \to I$

$$\sum_{i \in I} s_i = \sum_{j \in J} s_{\sigma(j)}.$$

Compatibility with the finite: For each finite index set $I = \{i_0, \dots, i_n\}$

$$\sum_{i \in I} s_i = s_{i_0} + \dots + s_{i_n}.$$

Partition invariance is actually a very strong property which, in particular, implies bijection invariance. Indeed, consider the partition $(I_j)_{j \in J}$ of I into singleton sets $I_j = \{\sigma(j)\}$. Then partition invariance (together with compatibility with finite sums) implies that $\sum_{i \in I} s_i = \sum_{j \in J} \sum_{i \in I_j} s_i = \sum_{j \in J} s_{\sigma(j)}$. Bijection invariance also justifies that we consider operations over index *sets* rather than, for instance, transfinite sequences.

Most natural infinitary operations on monoids satisfy these properties. Nevertheless there are quite simple constructions that violate, for instance, infinite associativity, even for certain naturally ordered monoids with an infinite sum defined as the supremum of its finite subsums.

▶ **Example 2.** Let $S = \mathbb{N} \cup \{\omega, \omega'\}$ with the (commutative) addition that extends the natural addition on $\mathbb{N}$ by $n + \omega = \omega$ for $n \in \mathbb{N}$ and $\omega + \omega = \omega + \omega' = \omega' + \omega' = \omega'$. Defining $\sum_{i \in I} s_i = \sup\{\sum_{i \in I_0} s_i \;:\; I_0 \subseteq^{\text{fin}} I\}$ we have a summation operator where, for the sequence $(s_n)_{n \in \mathbb{N}}$ with $s_0 = \omega$ and $s_n = 1$ for $n > 0$, we have that the finite sum takes are all the values $n \leq \omega$, and hence

$$\sum_{n \in \mathbb{N}} s_n = \sup\{n : n \leq \omega\} = \omega \quad \text{but} \quad s_0 + \sum_{n \geq 1} s_n = s_0 + \sup\{n : n < \omega\} = \omega + \omega = \omega',$$

so partition invariance fails.

3.2 Compactness and its Consequences

However, these three requirements do not suffice to avoid "pathological" definitions with undesirable behaviour. Consider, for instance, the monoid $(\mathbb{N} \cup \{\infty\}, +, 0)$ with the infinitary sum defined by $\sum_{i \in I} s_i = \infty$ for all infinite I (and satisfying compatibility with $+$ for finite index sets). This violates, for instance, the following two natural properties.

Neutrality: $\sum$ *respects the neutral element* if $\sum_{i \in I} s_i = \sum_{i \in I, s_i \neq 0} s_i$.
Idempotence: $\sum$ *respects idempotent elements* if for all $s \in S$ such that $s + s = s$, also $\sum_{i \in I} s = s$ for every index set $I \neq \varnothing$.

To guarantee these, and other, desirable properties, we propose *compactness* properties which essentially say that if the infinitary operation takes different values on two sequences $(s_i)_{i \in I}$ and $(t_j)_{j \in J}$ then this is already witnessed by finite subsets, in the sense that some finite subsequence of one takes a value that is not assumed by any finite subsequence of the other. More formally:

Compactness: The operator $\sum$ is *compact* if for all $(s_i)_{i \in I}$ and $(t_j)_{j \in J}$ we have that

$$\sum_{i \in I} s_i = \sum_{j \in J} t_j \quad \text{whenever} \quad \Big\{ \sum_{i \in I_0} s_i : I_0 \subseteq^{\text{fin}} I \Big\} = \Big\{ \sum_{j \in J_0} t_j : J_0 \subseteq^{\text{fin}} J \Big\}.$$

Strong compactness: The operator $\sum$ is *strongly compact* if for all $(s_i)_{i \in I}$ and $(t_j)_{j \in J}$ and all s, t we have that

$$s + \sum_{i \in I} s_i = t + \sum_{j \in J} t_j \quad \text{whenever} \quad \Big\{ s + \sum_{i \in I_0} s_i : I_0 \subseteq^{\text{fin}} I \Big\} = \Big\{ t + \sum_{j \in J_0} t_j : J_0 \subseteq^{\text{fin}} J \Big\}.$$

▶ **Lemma 3.**
1. *Every compact operator respects idempotent elements.*
2. *If a partition invariant operator respects idempotent elements then it also respects the neutral element.*
3. *If $\sum$ is partition invariant and respects idempotent elements, then there exists, for every $s \in \mathcal{S}$, a unique element $\infty \cdot s := \sum_{i \in I} s$ for every infinite I.*
4. *If $\sum$ is strongly compact, and $s + p = s$ then $s + \sum_{i \in I} p = s$ for every index set I.*

Proof. (1) If $s+s=s$ then for each index set I, the values $\sum_{i\in I_0} s$ for finite $I_0 \subseteq I$ are s and 0. Taking $J=\{0,1\}$ and $t_0=0$ and $t_1=s$ we also have s and 0 as values for subsequences. Thus, by compactness, $\sum_{i\in I} s = t_0+t_1 = 0+s = s$.

For (2) we note that since 0 is an idempotent element, we have that $\sum_{j\in J} 0 = 0$ for all index sets J. It follows by partition invariance that

$$\sum_{i\in I} s_i = \sum_{\substack{i\in I\\ s_i\neq 0}} s_i + \sum_{\substack{i\in I\\ s_i=0}} 0 = \sum_{\substack{i\in I\\ s_i\neq 0}} s_i.$$

For (3) we first observe that the bijection invariance of an infinitary summation operator implies that $\sum_{i\in I} s = \sum_{j\in J} s$ for every $s\in\mathcal{S}$ and all index sets I, J of the same cardinality. Hence there exists, for every $s\in\mathcal{S}$ and every infinite cardinal κ, a unique element $\kappa\cdot s = \sum_{i\in I} s$ for every index set I of cardinality κ. Obviously, $\omega\cdot s$ is idempotent, and we can decompose any index set of size κ into a partition of sets of size ω. By partition invariance, and the respect of idempotent elements, it follows that $\kappa\cdot s = \omega\cdot s =: \infty\cdot s$. To prove (4), we note that $s+p=s$ implies that $s+\sum_{i\in I_0} p = s$ for all finite I_0. With $J=\varnothing$ and $t=s$, strong compactness implies that $s+\sum_{i\in I} p = s$ for every index set I. ◀

But compactness also has consequences for finite sums. Recall that a finite monoid $\mathcal{S}=(S,+,0)$ is aperiodic if for every $s\in\mathcal{S}$ there exists some $n\in\mathbb{N}$ such that $(n+1)s = ns$.

▶ **Lemma 4.** *The compactness property, even just for finite sums, in a finite monoid $\mathcal{S}=(S,+,0)$ implies that $\mathcal{S}$ must be aperiodic.*

Proof. If $\mathcal{S}$ is not aperiodic then there exist $s\in S$ and some minimal $n\in\mathbb{N}$ such $ns\neq (n+1)s = ks$ for some $k<n$. But then the sums $\sum_{i=1}^{n} s$ and $\sum_{i=1}^{n+1} s$ have different values although they have the same sets of values for subsums, namely $\{0,s,2s,\dots,ns\}$, which contradicts compactness. ◀

We further notice that the existence of an infinitary operation with (some of) these properties can also have implications for the purely finitary properties of the monoid $\mathcal{S}=(S,+,0)$. Recall that $\mathcal{S}$ is $+$-positive if $s+t=0$ only holds for $s=t=0$. Further, $(S,+,0)$ is naturally ordered, if $s\leq t :\Leftrightarrow \exists r(s+r=t)$ is a partial order. Since $\leq$ is always reflexive and transitive this is the case if, and only if, $\leq$ is antisymmetric, i.e. $s\leq t$ and $t\leq s$ imply that $s=t$. Obviously, a naturally ordered monoid is $+$-positive, but the converse is not true.

▶ **Lemma 5.** *If $\mathcal{S}=(S,+,0)$ admits a partition invariant infinitary sum that respects the neutral element, then $\mathcal{S}$ is $+$-positive. If this sum is strongly compact then $\mathcal{S}$ is naturally ordered.*

Proof. Suppose that $s+t=0$. Since $\sum$ is partition invariant and respects the neutral element we have that $0=\sum_{i\in\mathbb{N}}(s+t) = s+\sum_{i\in\mathbb{N}}(t+s) = s+0 = s$. For the second claim, suppose that $s+r=t$ and $t+q=s$. For $p=r+q$ we thus have that $s+p=s$ and $t+p=t$. We have to prove that $s=t$. By Lemma 3, strong compactness implies that $s+\sum_{i\in\mathbb{N}} p = s$ and $t+\sum_{i\in\mathbb{N}} p = t$. But then, partition invariance implies that

$$s = s+\sum_{i\in\mathbb{N}} p = s+\sum_{i\in\mathbb{N}}(r+q) = (s+r)+\sum_{i\in\mathbb{N}}(q+r) = t+\sum_{i\in\mathbb{N}} p = t. \qquad ◀$$

▶ **Lemma 6.** *There is a monoid $\mathcal{S}=(S,+,0)$ that admits a partition-invariant and compact infinitary sum such that $\mathcal{S}$ is* not *naturally ordered and the sum therefore violates strong compactness. In particular, compactness does not imply strong compactness.*

Proof sketch. Consider the monoid $M := (\mathbb{N}^4 \cup \{\infty\}, +, 0)$ with $a + \infty = \infty$ for all a where the infinitary sum is defined by $\sum_{i \in I} s_i = \infty$ if there are infinitely many nonzero summands $s_i \neq 0$, and otherwise, $\sum_{i \in I} s_i$ corresponds to the usual finite sum of all (finitely many) nonzero summands. Clearly, M is naturally ordered by the usual component-wise partial order on $\mathbb{N}^4$ with an adjoined top element ∞. Moreover, the infinitary sum is both partition-invariant and bijection-invariant.

To construct $\mathcal{S}$, we set $a := (1, 0, 0, 0)$, $b := (0, 1, 0, 0)$, $c := (0, 0, 1, 0)$ and $d := (0, 0, 0, 1)$, and we identify $a + c \sim b$ and $b + d \sim a$. Let $\sim$ be the minimal congruence relation on M that satisfies this and set $\mathcal{S} := M/{\sim}$. It can be shown that such a congruence relation exists and that it satisfies $a \not\sim b$ and is compatible with the infinitary sum on M. Moreover, the infinitary sum on $\mathcal{S}$ inherits partition-invariance from the infinitary sum on M.

Further, it is possible to show that the infinitary sum on $\mathcal{S}$ is still compact. However, $\mathcal{S}$ is not naturally ordered, since $a \not\sim b$ implies $[a]_\sim \neq [b]_\sim$, but by definition, we have $[a]_\sim \leq [b]_\sim$ and $[b]_\sim \leq [a]_\sim$ due to $[a]_\sim + [c]_\sim = [a + c]_\sim = [b]_\sim$ and $[b]_\sim + [d]_\sim = [b + d]_\sim = [a]_\sim$. With Lemma 5, it follows that the infinitary sum on $\mathcal{S}$ cannot be strongly compact. ◀

Although compactness and strong compactness of an infinitary operation are powerful and convenient properties, there is the problem that they are not always easy to verify, and that there are relevant semirings where multiplication is not compact, as for instance $\mathbb{N}^\infty[\![X^\infty]\!]$, an infinitary extension of $\mathbb{N}[X]$ to be defined in Section 5, which does not even respect idempotent elements. We therefore will work with the following simpler property, which by Lemma 3 is implied by compactness, is easier to establish, and suffices for our proofs.

Unique infinite powers: $\sum$ *has unique infinite powers* if for every $s \in \mathcal{S}$, there exists a unique element $\infty \cdot s$ with $\infty \cdot s := \sum_{i \in I} s$ for every infinite I.

3.3 Distributivity

The requirements that relate the two algebraic operations in a commutative semiring are the distributive law $s(r + t) = sr + st$ and the fact that the neutral element of addition is multiplicatively annihilating, i.e. $0 \cdot s = 0$ for all s. If an infinitary product $\prod$ is partition invariant and compatible with finite products, then it follows immediately that $\prod_{i \in I} s_i = 0$ whenever $s_i = 0$ for some $i \in I$. The generalisation of the distributive law to infinitary operations is more complicated and comes in a weak and a strong variant:

Weak distributivity: For each index set I and all s

$$s \cdot \sum_{i \in I} s_i = \sum_{i \in I} (s \cdot s_i).$$

Strong distributivity: For every index set I and every collection $(J_i)_{i \in I}$ of index sets

$$\prod_{i \in I} \sum_{j \in J_i} s_j = \sum_{f \in F} \prod_{i \in I} s_{f(i)},$$

where F is the set of all choice functions $f \colon I \to \bigcup_{i \in I} J_i$ such that $f(i) \in J_i$ for all $i \in I$.

For finite index sets I, strong distributivity is implied by weak distributivity via a rather straightforward induction. For infinite index sets the situation is more complicated.

We first observe that strong distributivity holds for the completion $\mathbb{N}^\infty = \mathbb{N} \cup \{\infty\}$ of the natural semiring $(\mathbb{N}, +, \cdot, 0, 1)$ (with the natural extensions of finite addition and multiplication from $\mathbb{N}$ to finite and infinitary addition and multiplication on $\mathbb{N}^\infty$).

▶ **Proposition 7.** *The semiring $\mathbb{N}^\infty$ satisfies strong distributivity.*

Proof. Given an expression $\prod_{i\in I}\sum_{j\in J_i} s_j$ over $\mathbb{N}^\infty$, we argue by case distinction.

If $\prod_{i\in I}\sum_{j\in J_i} s_j = 0$, then there is some $i \in I$ such that $s_j = 0$ for all $j \in J_i$. But then every choice function $f \in F$ has the property that $s_{f(i)} = 0$ which implies that also $\sum_{f\in F}\prod_{i\in I} s_{f(i)} = 0$.

If $\prod_{i\in I}\sum_{j\in J_i} s_j \neq 0$, then there exists, for each $i \in I$, some $j \in J_i$ such that $s_j \neq 0$. Hence there is a choice function $f \in F$ such that $\prod_{i\in I} s_{f(i)} \neq 0$.

We next discuss the cases where $\prod_{i\in I}\sum_{j\in J_i} s_j = \infty$. Assume that $s_k = \infty$ for some $k \in J_i$, so $\sum_{j\in J_i} s_j = \infty$. For the choice function that selects $k \in J_i$, and nonzero elements in the other index sets, we have $\prod_{i\in I} s_{f(i)} = \infty$, and hence $\sum_{f\in F}\prod_{i\in I} s_{f(i)} = \infty$. Another possible case with $\sum_{j\in J_i} s_j = \infty$ appears when there are infinitely many $j \in J_i$ with $s_j \neq 0$. Then there are infinitely many choice functions $f \in F$ such that $s_{f(i)} \neq 0$ for all i, so again $\sum_{f\in F}\prod_{i\in I} s_{f(i)} = \infty$. Finally it may be the case that $\prod_{i\in I}\sum_{j\in J_i} s_j = \infty$ because there are infinitely many $i \in I$, such that $\sum_{j\in J_i} s_j > 1$. We distinguish two possibilities. Either there are infinitely many i, for which there exists some $j \in I_j$ with $s_j \geq 2$. Selecting these elements, and (by the argument above) non-zero values for the other indices gives us a choice function $f \in F$ such that $\prod_{i\in I} s_{f(i)} = \infty$. The other possibility is that there exist infinitely many $i \in I$ with at least two indices $j \in I_j$ with $s_j = 1$. But this implies that there are not just one, but infinitely many choice functions $f \in F$ such that $s_{f(i)} \neq 0$ for all i, so $\sum_{f\in F}\prod_{i\in I} s_{f(i)} = \infty$.

Suppose finally that $\prod_{i\in I}\sum_{j\in J_i} s_j = n$ where $1 \leq n < \infty$. Then there exists a finite index set $I_0 \subset I$, such that $\prod_{i\in I}\sum_{j\in J_i} s_j = \prod_{i\in I_0}\sum_{j\in J_i} s_i$ and that $\sum_{j\in J_i} s_j = 1$ for all $i \in I \setminus I_0$. This implies that in each such J_i there is precisely one j such that $s_j = 1$, and that $s_k = 0$ for $k \neq j$. Let F' be the subset of those choice functions in F that select for each $i \in I \setminus I_0$ the unique $j \in J_i$ with $s_j = 1$. Notice that $\prod_{i\in I} s_{f(i)} = 0$ for all $f \in F \setminus F'$. Further, let F_0 be the set of choice functions on I_0. Each $f \in F_0$ uniquely extends to a choice function in $f' \in F'$, with $\prod_{i\in I_0} s_{f(i)} = \prod_{i\in I} s_{f'(i)}$. We further note that each sum $\sum_{j\in J_i} s_j$ cannot exceed n, so it can only have finitely many non-zero entries and can be written as a finite sum. By (finite) distributivity, we have

$$\sum_{f\in F}\prod_{i\in I} s_{f(i)} = \sum_{f'\in F'}\prod_{i\in I} s_{f'(i)} + \sum_{f\in F\setminus F'}\prod_{i\in I} s_{f(i)} = \sum_{f\in F_0}\prod_{i\in I_0} s_{f(i)} = \prod_{i\in I_0}\sum_{j\in J_i} s_j = n. \qquad ◀$$

It is easy to verify that $\mathbb{N}^\infty$ also satisfies all other properties mentioned above, including (strong) compactness. The same holds for other semirings that are obtained by completing a semiring without infinitary operations by an element ∞ to which the appropriate infinite sums and infinite products evaluate. This includes, for instance, the polynomial semirings $\mathbb{N}[X] \cup \{\infty\}$ and $\mathbb{B}[X] \cup \{\infty\}$, for finite sets X of indeterminates.

But there are also important semirings for which strong distributivity depends on the cardinality of the index sets that we consider. We illustrate this for the tropical semiring $\mathbb{T} = (\mathbb{R}_+^\infty, \min, +, \infty, 0)$ (whose infinitary operations are, of course, infimum and the natural infinitary sum). Weak distributivity holds for arbitrary index sets. However, this is not the case for strong distributivity.

▶ **Proposition 8.** *In the tropical semiring, strong distributivity holds for countable index sets, but fails for uncountable ones.*

Proof. We first prove the failure of strong distributivity for uncountable index sets. Let $J_i = \omega$ for all i in some uncountable index set I, and let $(s_j)_{j\in\omega}$ be any sequence of positive real numbers that converges to 0. Strong distributivity would mean that

$$\sum_{i\in I}\inf_{j\in\omega} s_j = \inf_{f\in F}\sum_{i\in I} s_{f(i)}$$

where F ranges over all choice functions $f\colon I \to \omega$. However, the left side is 0 since the infimum of $(s_j)_{j\in\omega}$ is 0. But every choice function $f\colon I \to \omega$ must hit some $n \in \omega$ infinitely often so for every f we have that $\sum_{i\in I} s_{f(i)} \geq \infty \cdot s_n = \infty$. Hence the right side evaluates to ∞, and strong distributivity fails.

Next we observe that, for all index sets I and $(J_i)_{i\in I}$,

$$\sum_{i\in I} \inf_{j\in J_i} s_j \leq \inf_{f\in F} \sum_{i\in I} s_{f(i)}.$$

Indeed, for all $f \in F$ we clearly have that $\inf_{j\in J_i} s_j \leq s_{f(i)}$. Summation is monotone, so it follows that $\sum_{i\in I} \inf_{j\in J_i} s_j \leq \sum_{i\in I} s_{f(i)}$ and since this holds for all $f \in F$ it also holds for the infimum.

Finally, it remains to show that for *countable* I and all $(J_i)_{i\in I}$, we have that

$$\sum_{i\in I} \inf_{j\in J_i} s_j \geq \inf_{f\in F} \sum_{i\in I} s_{f(i)}.$$

We know that this holds for finite I, so we assume now that I is countably infinite. Without loss of generality we can take $I = \omega$, and we set $q_i := \inf_{j\in I_j} s_j$ and $q := \sum_{i\in\omega} q_i$. We have to show that $\inf_{f\in F} \sum_{i\in\omega} s_{f(i)} \leq q$. If $q = \infty$, there is nothing to prove. Otherwise q and hence also all q_i are finite. Fix any $\varepsilon > 0$. Since $\inf_{j\in J_i} s_j = q_i$ we can find, for every $i < \omega$, some $j \in J_i$ such that $s_j \leq q_i + \varepsilon 2^{-(i+1)}$. Let f_ε be a choice function that maps each $i \in \omega$ to some $j \in J_i$ with this property. We then have that

$$\sum_{i\in\omega} s_{f_\varepsilon(i)} \leq \sum_{i\in\omega} (q_i + \varepsilon 2^{-(i+1)}) = \sum_{i\in\omega} q_i + \varepsilon \cdot \sum_{i\in\omega} 2^{-(i+1)} = q + \varepsilon.$$

Since this holds for all $\varepsilon > 0$ we conclude that $\inf_{f\in F} \sum_{i\in I} s_{f(i)} \leq q$. ◀

The same proposition, with almost exactly the same proofs, holds also for the Viterbi semiring $\mathbb{V}$, the Łukasiewicz semiring $\mathbb{L}$ and the semiring of doubt $\mathbb{D}$.

3.4 Monotonicity

In a naturally ordered semiring, an important property is that both addition and multiplication are *monotone* in each argument: if $s_1 \leq t_1$ and $s_2 \leq t_2$ then $s_1 + s_2 \leq t_1 + t_2$ and $s_1 s_2 \leq t_1 t_2$. By partition invariance it immediately follows that also the infinitary sum is monotone in each argument.

▶ **Lemma 9.** *If $s_i \leq t_i$ for all $i \in I$, then $\sum_{i\in I} s_i \leq \sum_{i\in I} t_i$.*

Proof. For each $i \in I$, we have that $t_i = s_i + \delta_i$ for some element $\delta_i \in S$. Hence $\sum_{i\in I} t_i = \sum_{i\in I}(s_i + \delta_i) = \sum_{i\in I} s_i + \sum_{i\in I} \delta_i$. ◀

Monotonicity of multiplication is implied by the distributive law: if $t_1 = s_1 + \delta_1$ and $t_2 = s_2 + \delta_2$ then $t_1 t_2 = s_1 s_2 + s_1 \delta_2 + s_2 \delta_1 + \delta_1 \delta_2$, so $s_1 s_2 \leq t_1 t_2$. Monotonicity of infinitary products in each single argument follows by the same argument. If $t_j = s_j + \delta$ for some $j \in I$, and $s_i = t_i$ for all other $i \in I$ then $\prod_{i\in I} t_i = (s_j + \delta) \prod_{i\in I\setminus\{j\}} s_i = \prod_{i\in I} s_i + \delta \prod_{i\in I\setminus\{j\}} s_i$. Assuming strong distributivity, we can apply this argument simultaneously to each factor.

▶ **Lemma 10.** *If $\mathcal{S}$ satisfies strong distributivity, and $s_i \leq t_i$ for all $i \in I$, then $\prod_{i\in I} s_i \leq \prod_{i\in I} t_i$.*

Proof. Again, let $t_i = s_i + \delta_i$ and let F be the set of all choice functions f that map each $i \in I$ to either s_i or δ_i. Further, let $f_0 \in F$ be the function that maps all $i \in I$ to s_i. By strong distributivity, we have that

$$\prod_{i\in I} t_i = \prod_{i\in I}(s_i+\delta_i) = \sum_{f\in F}\prod_{i\in I} f(i) = \prod_{i\in I} f_0(i) + \sum_{f\in F\setminus\{f_0\}}\prod_{i\in I} f(i) = \prod_{i\in I} s_i + \sum_{f\in F\setminus\{f_0\}}\prod_{i\in I} f(i)$$

so $\prod_{i\in I} s_i \leq \prod_{i\in I} t_i$. ◀

Given that strong distributivity does not hold in all interesting semirings, we should add monotonicity to our list of desired properties for infinitary semiring operations.

Monotonicity: For each index set I and all families $(s_i)_{i\in I}$ and $(t_i)_{i\in I}$ such that $s_i \leq t_i$ (w.r.t. the natural order) we also have that

$$\sum_{i\in I} s_i \leq \sum_{i\in I} t_i \qquad \text{and} \qquad \prod_{i\in I} s_i \leq \prod_{i\in I} t_i.$$

The requirement of monotonicity for the infinitary sum is redundant since it is implied by partition invariance, but monotonicity of infinitary products does not seem to follow from weaker properties than strong distributivity.

▶ **Example 11.** Let $S = \mathbb{N} \cup \{\infty\}$ with the natural definition of infinitary sum, but with an infinitary product that evaluates to 0, if there are infinitely many finite factors different from 1. More precisely,

$$\prod_{i\in I} s_i = \begin{cases} \infty & \text{if } s_i = \infty \text{ for some } i \in I \text{ and } s_i \neq 0 \text{ for all } i \in I \\ \prod_{i\in I_0} s_i & \text{for } I_0 \subseteq^{\text{fin}} I \text{ such that } s_i = 1 \text{ for all } i \in I \setminus I_0 \\ 0 & \text{otherwise.} \end{cases}$$

The infinitary product is not monotone since $\prod_{i<\omega} 1 = 1$ but $\prod_{i<\omega} 2 = 0$. Strong distributivity of course also fails, as witnessed by $\prod_{i<\omega}(1+1)$. One can readily verify that the infinitary operations in this semiring satisify all the other properties that we discussed, including strong compactness.

3.5 Infinitary Semirings

We are now ready to propose a definition for semirings with infinitary operations.

▶ **Definition 12.** *An* infinitary semiring*, also called ∞-semiring, is a commutative, naturally ordered semiring $\mathcal{S} = (S, +, \cdot, 0, 1)$, together with two infinitary operations $\sum$ and $\prod$ that satisfy the following properties:*

- *partition invariance (infinite associativity), and hence also bijection invariance (infinite commutativity),*
- *compatibility with finite addition and multiplication,*
- *neutral elements are respected,*
- *there are unique infinite powers,*
- *weak distributivity, and*
- *monotonicity.*

This should be seen as a working definition that reflects our current state of investigations. There are of course alternative possibilities. For instance we could impose (strong) compactness of infinitary sums and products as a basic property, which would imply that idempotent and neutral elements are respected, and that there exist unique infinite powers. We further note that the requirement that the additive neutral element is respected is in fact redundant, as it is implied by weak distributivity and partition invariance. Indeed, $\sum_{i\in I} 0 = \sum_{i\in I}(0\cdot 0) = 0\cdot\sum_{i\in I} 0 = 0$ and therefore $\sum_{i\in I} s_i = \sum_{i\in I, s_i=0} s_i + \sum_{i\in I, s_i\neq 0} s_i = \sum_{i\in I, s_i\neq 0} s_i$. However, it can be shown that the requirement that the multiplicative neutral element is respected by infinitary products does not follow from the other properties (for a counterexample, take $(\mathbb{N}^\infty, \cdot, 1)$ and set all infinite products of non-zero values to ∞). We could also require strong distributivity, which we chose to omit from the definition because we want to include in our study some relevant semirings that (for arbitrary index sets) only satisfy the weak distributive law. Instead, we introduce the following variant.

▶ **Definition 13.** *Let κ be an infinite cardinal. An infinitary semiring is* κ-distributive *if it satisfies strong distributivity for products of cardinality $<\kappa$. That is, it satisfies*

$$\prod_{i\in I}\sum_{j\in J_i} s_j = \sum_{f\in F}\prod_{i\in I} s_{f(i)}$$

for all sets I with $|I|<\kappa$ (and sets J_i of arbitrary cardinality). It is strongly distributive *if it satisfies strong distributivity for all index sets.*

The tropical semiring, the Viterbi semiring, and the Łukasiewicz semiring are examples of ω_1-distributive semirings.

Homomorphisms between ∞-semirings should be compatible with the infinitary operations. We again introduce a weaker variant for a more fine-grained analysis.

▶ **Definition 14** (∞-semiring homomorphisms)**.** *Let κ be an infinite cardinal, and let $\mathcal{S},\mathcal{S}'$ be infinitary semirings. A* semiring κ-homomorphism $h\colon \mathcal{S}\to\mathcal{S}'$ *is a semiring homomorphism such that for all sequences $(s_i)_{i\in I}$ in $\mathcal{S}$ with $|I|<\kappa$, we have that*

$$h\Big(\sum_{i\in I} s_i\Big) = \sum_{i\in I} h(s_i) \qquad \text{and} \qquad h\Big(\prod_{i\in I} s_i\Big) = \prod_{i\in I} h(s_i).$$

Further h is called an ∞-semiring homomorphism*, if it is a semiring κ-homomorphism for all κ.*

Of course, the term *infinitary semiring* does not imply that the semiring has infinitely many elements. Quite to the contrary:

▶ **Proposition 15.** *Every finite semiring, in which both addition and multiplication induce aperiodic monoids, expands to an ∞-semiring (in which, moreover, both operations are strongly compact).*

Proof. Let $(S,\cdot,1)$ be the multiplicative monoid of the semiring. Since it is aperiodic, there exists, for every $s\in S$, a minimal number n_s such that $s^{n_s+1}=s^{n_s}$ and hence $s^n = s^{n_s}$ for all $n\geq n_s$. We put $s^\infty := s^{n_s}$. We can now define an infinitary product $\prod_{i\in I} s_i$ by reducing it to a finite product. For every $s\in S$, let $m_s := \min(n_s, |\{i\in I : s_i = s\}|)$ and set $\prod_{i\in I} s_i := \prod_{s\in S} s^{m_s}$. The definition of infinitary sums is completely analogous. It is easily verified, that the required properties are inherited from finite addition and multiplication. Since the infinitary operations reduced to the finite ones, strong compactness is also straightforward. ◀

Let us next consider infinite lattice semirings, induced by some partial order $(S, \leq)$.

▶ **Proposition 16.** *An infinite lattice semiring expands to an ∞-semiring if, and only if, the underlying order is a complete lattice in which finite infima distribute over arbitrary suprema.*

Proof. In a complete lattice semiring where finite infima distribute over arbitrary suprema, the desired infinitary operations are suprema and infima, which obviously satisfy all required properties. For the converse implication, it suffices to show that in any expansion to an ∞-semiring, infinitary summation is given by suprema. As required, this implies the existence of arbitrary suprema and thus completeness of the lattice, and the weak distributive law w.r.t. the semiring operations reduces to weak distributivity of the lattice operations. Suppose that there are elements $(x_i)_{i\in I}$ such that $z := \sum_{i\in I} x_i$ is not the supremum of $\{x_i \mid i \in I\} =: X$. By partition invariance, $x_i \sqcup \sum_{j\in I, j\neq i} x_i = z$ for each $i \in I$, so z is an upper bound of X. Thus, there must exist some other upper bound y for X such that $z \not\leq y$. But weak distributivity yields $y \sqcap z = y \sqcap \sum_{i\in I} x_i = \sum_{i\in I}(y \sqcap x_i) = \sum_{i\in I} x_i = z$ and thus $z \leq y$, a contradiction. ◀

There are two main further classes of semirings that we consider. A particularly useful class is the class of infinitary absorptive semirings, studied in the next section. Recall that absorption has the consequence that multiplication is decreasing, which leads to dualities that permit to carry over a number of classical logical properties to semiring semantics. The other relevant class consists of the semirings that extend the natural semiring $\mathbb{N}$ or semirings of polynomials such as $\mathbb{N}[X]$, where multiplication is increasing.

4 Infinitary Absorptive Semirings

Recall that a semiring $\mathcal{S}$ is absorptive if $s + st = s$ for all $s, t \in \mathcal{S}$ or, equivalently, $1 + t = 1$ for all $t \in \mathcal{S}$. Every absorptive semiring is idempotent (i.e., $s + s = s$ for all $s \in \mathcal{S}$) and every idempotent semiring is naturally ordered. In naturally ordered semirings, addition and multiplication are monotone w.r.t. the natural order. Further in absorptive semirings, multiplication is decreasing (i.e., $s \cdot r \leq s$ for all $s, r \in \mathcal{S}$).

We will introduce the notion of infinitary absorptive semirings. These are based on absorptive semirings, with some additional properties that permit to define natural infinitary addition and multiplication operations, based on infima and suprema.

▶ **Definition 17.** *An* infinitary absorptive semiring *is the expansion of an absorptive semiring $\mathcal{S}$ which satisfies the additional properties that*

- *the natural order $(\mathcal{S}, \leq)$ is a complete lattice.*
- *$\mathcal{S}$ is (fully) continuous: for every non-empty chain $C \subseteq S$, the supremum $\bigsqcup C$ and the infimum $\bigsqcap C$ are compatible with addition and multiplication, i.e.*

$$s \circ \bigsqcup C = \bigsqcup (s \circ C) \quad \text{and} \quad s \circ \bigsqcap C = \bigsqcap (s \circ C),$$

where $(s \circ C) := \{s \circ c : c \in C\}$ for every $s \in S$ and $\circ \in \{+, \cdot\}$.

As a consequence, we can define natural infinitary addition and multiplication operations in $\mathcal{S}$, by taking suprema of finite subsums and infima of finite subproducts:

$$\sum_{i\in I} s_i := \bigsqcup_{\substack{I_0\subseteq I\\ I_0 \text{ finite}}} \Big(\sum_{i\in I_0} s_i\Big) \quad \text{and} \quad \prod_{i\in I} s_i := \bigsqcap_{\substack{I_0\subseteq I\\ I_0 \text{ finite}}} \Big(\prod_{i\in I_0} s_i\Big).$$

Since addition is idempotent in absorptive semirings, the infinitary addition is in fact the same as the supremum: $\sum_{i\in I} s_i = \bigsqcup_{i\in I} s_i$. However, unless multiplication is also idempotent (so that the semiring is a lattice semiring), infinitary products need not coincide with infima. Indeed, we note that there are infinitary absorptive semirings in which the natural order is a completely distributive lattice (i.e., infima and suprema satisfy a strong distributive law), but strong distributivity does not hold for the infinitary semiring operations defined above. One such example is the Viterbi semiring $\mathbb{V}$: the natural order for $\mathbb{V}$ is just the usual linear order on the real interval $[0, 1]$, which is a completely distributive lattice, but we have seen that the strong distributive law fails on $\mathbb{V}$ for uncountable index sets.

Most of the common application semirings mentioned in Sect. 2 are in fact absorptive (with the notable exception of the natural semiring) and permit the expansion to an infinitary absorptive semiring. Among the semirings of polynomials mentioned in Sect. 2, only $\mathbb{S}(X)$ and $\mathrm{PosBool}(X)$ are absorptive semirings whereas $\mathbb{N}[X]$, $\mathbb{B}[X]$, and $\mathbb{W}(X)$ are not.

It remains to show that infinitary absorptive semirings are indeed ∞-semirings, i.e. that they satisfy all the properties required by Definition 12. Since the infinitary properties are based on suprema and infima, this is straightforward in most cases. For the weak distributivity law, this is a direct consequence of the continuity of multiplication. The only property that requires work, and also makes use of continuity of multiplication, is the partition invariance of infinitary products (whereas for infinite sums partition invariance is trivial, because they are just suprema).

▶ **Lemma 18.** *Products in infinitary absorptive semirings are partition invariant.*

Proof. To simplify notation, we define the abbreviation $s(I_0) := \prod_{i\in I_0} s_i$ for *finite* index sets $I_0 \subseteq^{\text{fin}} I$. We thus have to prove that

$$\prod_{i\in I} s_i = \bigsqcap_{I_0\subseteq^{\text{fin}} I} s(I_0) \overset{!}{=} \bigsqcap_{J_0\subseteq^{\text{fin}} J} \Big(\prod_{j\in J_0} \bigsqcap_{H_0\subseteq^{\text{fin}} I_j} s(H_0)\Big) = \prod_{j\in J}\prod_{i\in I_j} s_i .$$

We prove both directions. First fix a finite set $I_0 \subseteq^{\text{fin}} I$. Since $(I_j)_{j\in J}$ is a partition, there is a finite set $J_0 \subseteq^{\text{fin}} J$ such that $I_0 \subseteq \bigcup_{j\in J_0} I_j$. Moreover, for each $i \in I_j$ we clearly have $s_i \geq \bigsqcap_{H_0\subseteq^{\text{fin}} I_j} s(H_0)$ by considering $H_0 = \{i\}$. Using absorption (abs) and monotonicity (m), we have

$$s(I_0) = \prod_{i\in I_0} s_i \overset{\text{(abs)}}{\geq} \prod_{\substack{i\in I_j\\ j\in J_0}} s_i \overset{\text{(m)}}{\geq} \prod_{j\in J_0} \bigsqcap_{H_0\subseteq^{\text{fin}} I_j} s(H_0) \geq \bigsqcap_{J_0\subseteq^{\text{fin}} J}\prod_{j\in J_0} \bigsqcap_{H_0\subseteq^{\text{fin}} I_j} s(H_0)$$

which proves direction "$\geq$".

For the other direction, fix a finite set $J_0 = \{j_1, \dots, j_k\} \subseteq^{\text{fin}} J$. Recall that $s(I_0)$ is a finite product and thus associative (a). Together with continuity of multiplication (c), we get

$$\begin{aligned}\bigsqcap_{I_0\subseteq^{\text{fin}} I} s(I_0) &\leq \bigsqcap_{I_0\subseteq^{\text{fin}} \bigcup_{j\in J_0} I_j} s(I_0) = \bigsqcap_{H_{j_1}\subseteq^{\text{fin}} I_{j_1}} \dots \bigsqcap_{H_{j_k}\subseteq^{\text{fin}} I_{j_k}} s(H_{j_1}\cup\dots\cup H_{j_k})\\ &\overset{\text{(a)}}{=} \bigsqcap_{H_{j_1}\subseteq^{\text{fin}} I_{j_1}} \dots \bigsqcap_{H_{j_k}\subseteq^{\text{fin}} I_{j_k}} (s(H_{j_1})\cdots s(H_{j_k}))\\ &\overset{\text{(c)}}{=} \Big(\bigsqcap_{H_{j_1}\subseteq^{\text{fin}} I_{j_1}} a_{H_{j_1}}\Big)\cdots\Big(\bigsqcap_{H_{j_k}\subseteq^{\text{fin}} I_{j_k}} a_{H_{j_k}}\Big) = \prod_{j\in J_0}\bigsqcap_{H_0\subseteq^{\text{fin}} I_j} s(H_0),\end{aligned}$$

which closes the proof. ◀

The natural notion of homomorphisms between infinitary absorptive semirings are the fully-continuous homomorphisms (which are compatible with suprema and infima of chains in the same way as the semiring homomorphisms). Since infinitary operations are defined through suprema and infima[1], they are preserved by fully-continuous homomorphisms.

▶ **Proposition 19.** *Every fully-continuous homomorphism $h\colon \mathcal{S} \to \mathcal{S}'$ between infinitary absorptive semirings is an ∞-semiring homomorphism.*

5 Polynomials and Power Series

The semirings $\mathbb{N}[X]$ of multivariate polynomials with a (finite) set X of indeterminates and coefficients from $\mathbb{N}$ play a fundamental role for the provenance analysis of database queries and first-order sentences. This is due to the fact that $\mathbb{N}[X]$ is the semiring that is freely generated by X and has the universal property that every function $h\colon X \to \mathcal{S}$ into an arbitrary commutative semiring $\mathcal{S}$ uniquely extends to a semiring homomorphism $h\colon \mathbb{N}[X] \to \mathcal{S}$.

The question arises, whether we can extend $\mathbb{N}[X]$ to an infinitary semiring that has corresponding universal properties. We must be able to infinitely often add the same monomial (e.g., $x+x+x+\dots$), add infinitely many different monomials (e.g., $x+x^2+x^3+\dots$) and multiply the same variable infinitely often (e.g., $x \cdot x \cdot x \cdot \dots$). To address these issues, we extend $\mathbb{N}[X]$ by allowing coefficients in $\mathbb{N}^\infty$, using formal power series instead of polynomials, and allowing exponents in $\mathbb{N}^\infty$ (as is done for generalised absorptive polynomials). We thus obtain *semirings of generalised power series over X*, denoted $\mathbb{N}^\infty[\![X^\infty]\!]$. Infinite summation in $\mathbb{N}^\infty[\![X^\infty]\!]$ is straightforward and infinite products can be defined by considering the sum over all possible factorisations of a given monomial. Here are the formal definitions.

▶ **Definition 20.** *Fix a finite set X of indeterminates. A* monomial *is a function $m\colon X \to \mathbb{N}^\infty$ that associates with each indeterminate an exponent. Let M be the set of all monomials.*

A generalised power series *is a function $P\colon M \to \mathbb{N}^\infty$ associating with each monomial its coefficient. We obtain the semiring $\mathbb{N}^\infty[\![X^\infty]\!]$ of generalised power series with the infinitary sum and product defined by*

$$\sum_{i\in I} P_i := \Big(m \mapsto \sum_{i\in I} P_i(m)\Big) \quad \textit{and} \quad \prod_{i\in I} P_i = \Big(m \mapsto \sum_{(m_i)_i \in \mathsf{splits}(m)} \prod_{i\in I} P_i(m_i)\Big).$$

Here, $\mathsf{splits}_I(m)$ is the set of sequences $(m_i)_{i\in I}$ of monomials with $m = \prod_{i\in I} m_i$. We may omit the index I if it is clear from the context.

The corresponding definition with an *infinite* set X of indeterminates is not consistent with Definition 12, since it does not have unique powers. For the polynomial $P = \sum_{i<\omega} x_i$, we have that P^ω is different from P^{ω_1}. Therefore, we only use finite sets of indeterminates. We further note that $\mathbb{N}^\infty[\![X^\infty]\!]$, even with a single indeterminate, does not preserve idempotent elements and hence is not compact. Indeed, let $Q := \sum_{i<\omega} \infty \cdot x^i$. Then $Q^2 = Q$ but $Q^\omega = \infty \cdot x^\infty + Q \neq Q$. Nevertheless $\mathbb{N}^\infty[\![X^\infty]\!]$ turns out to be an important ∞-semiring, playing a similar role as $\mathbb{N}[X]$ does in the finite case. To establish that $\mathbb{N}^\infty[\![X^\infty]\!]$ is an ∞-semiring, we begin with somewhat technical observations about infinite powers.

[1] Fully-continuous homomorphisms commute with suprema and infima of *chains* by definition, and thus also with suprema/infima of the *directed sets* in the definition (based on arguments in [9]).

▶ **Lemma 21.** *Let $P \in \mathbb{N}^\infty[\![X^\infty]\!]$ and I an infinite index set. The only possible coefficients of $\prod_{i\in I} P$ are 0, 1, and ∞.*

Proof. Let $Q = \prod_{i\in I} P$, fix a monomial m and assume that $Q(m) \neq 0$. Consider a sequence $(m_i)_{i\in I} \in \mathsf{splits}(m)$ with value $\prod_{i\in I} P(m_i) > 0$. If there is a monomial v that occurs only finitely often (but at least once) in $(m_i)_i$, then by permuting the occurrences of v with other monomials in the sequence, we obtain infinitely many pairwise different sequences in $\mathsf{splits}(m)$ with the same value. Since $Q(m)$ is the sum over all sequences in $\mathsf{splits}(m)$, this implies $Q(m) = \infty$. Similarly, if two different monomials v, v' occur infinitely often in $(m_i)_i$, we can again obtain infinitely many different permutations of the sequence and $Q(m) = \infty$.

So the only possibility for $Q(m) < \infty$ is that all sequences $(m_i)_i$ with $\prod_{i\in I} P(m_i) > 0$ consist of only one monomial, i.e. $m_i = m_j$ for all $i, j \in I$. Assume towards a contradiction that two such sequences exist, say $(v)_{i\in I} \in \mathsf{splits}(m)$ and $(w)_{i\in I} \in \mathsf{splits}(m)$. Since $|\omega| + |I| = |I|$, we can construct a sequence $(u_i)_{i\in I}$ that consists of countably many repetitions of v, and $|I|$ many repetitions of w. All indeterminates occurring in v or w must have exponent ∞ in m, hence also $(u_i)_i \in \mathsf{splits}(m)$. But this sequence uses two monomials infinitely often which implies $Q(m) = \infty$, contradiction.

Hence $Q(m) < \infty$ implies that there is only one sequence $(m_i)_i \in \mathsf{splits}(m)$ with value $\prod_{i\in I} P(m_i) > 0$, and since $m_i = m_j$ for all i, j this value must be either 1 or ∞. ◀

▶ **Lemma 22.** *$\mathbb{N}^\infty[\![X^\infty]\!]$ has unique infinite powers.*

Proof. We recall that $\mathbb{N}^\infty$ satisfies compactness and thus has unique infinite powers. The unique power property for summation in $\mathbb{N}^\infty[\![X^\infty]\!]$ is inherited from $\mathbb{N}^\infty$, as summation is defined pointwise by summing over the coefficients of each monomial.

Multiplication requires more work. Notice that the set M of monomials is countable, since X is finite. Let $P \in \mathbb{N}^\infty[\![X^\infty]\!]$ and I, J two infinite index sets. It suffices to prove that

$$Q \coloneqq \prod_{i\in I} P \;\leq\; \prod_{j\in J} P =: R$$

due to symmetry. Fix a monomial m and a sequence $(m_i)_{i\in I} \in \mathsf{splits}_I(m)$. Consider the set of monomials $\{m_i \mid i \in I\}$ occurring in this sequence. We partition this set into a set M_0 of monomials that occur only finitely often and a set M_∞ of monomials that occur infinitely often. Let $I_0 \subseteq I$ be the set of indices i with $m_i \in M_0$. Since M_0 is countable and each $m \in M_0$ occurs finitely often, I_0 is countable as well. We further fix an enumeration of the set M_∞ (which is countable as well).

We now construct a sequence $(v_j)_{j\in J} \in \mathsf{splits}_J(m)$ by first constructing a sequence $(w_l)_{l\in L} \in \mathsf{splits}_L(m)$ for some index set L and then applying a bijection $f\colon J \to L$. We define L as follows:

$$L = I_0 \;\dot\cup\; (\omega \times \omega) \;\dot\cup\; J.$$

We next define the elements of the sequence $(w_l)_{l\in L}$. We first need a "padding element" m_∞ (in case $|J| > |I|$). If $M_\infty \neq \varnothing$, we choose an arbitrary but fixed $m_\infty \in M_\infty$. If $M_\infty = \varnothing$, then M_0 contains infinitely many monomials (each of which occurs finitely often in $(m_i)_{i\in I}$). Recall that $m = \prod_{i\in I} m_i$ and consider the indeterminates $X_0 \subseteq X$ with *finite* exponents in m. Since the exponents are finite, there can be only finitely many monomials in M_0 containing an indeterminate in X_0 (recall that X is finite!). Consider the infinitely many monomials in M_0 *not* containing an indeterminate in X_0. Among these, we choose m_∞ so that $P(m_\infty)$ is minimal. Notice that when $P(m_\infty) > 1$, then by minimality there are infinitely many monomials $m' \in M_0$ with $P(m') > 1$, and hence $\prod_{i\in I} P(m_i) = \infty$ (†).

We are now ready to define the sequence $(w_l)_{l \in L}$.

- if $l \in I_0$, we set $w_l = m_l$ (i.e., we copy all finitely often occurring monomials),
- if $l = (n, m) \in \omega \times \omega$, we set w_l to the n-th monomial in M_∞ (i.e., we repeat each monomial in M_∞ infinitely often); if $M_\infty = \varnothing$ we set $w_l = m_\infty$ (or we omit the $(\omega \times \omega)$-part),
- if $l \in J$, we set $w_l = m_\infty$ (this is only for padding so that L has the right cardinality).

By construction, the monomials in $(w_l)_{l \in L}$ match the monomials in $(m_i)_{i \in I}$ in the following sense: each monomial appears either infinitely often in both sequences, or the same finite number of times in both sequences. The only exception is m_∞ in the case $M_\infty = \varnothing$, but in this case we know that m_∞ contains only indeterminates that have exponent ∞ in m. It follows that the products of the two sequences result in the same monomial m, so $(w_l)_{l \in L} \in \mathsf{splits}_L(m)$ as claimed.

It further follows (using compactness of $\mathbb{N}^\infty$) that the values of the sequences are also equal: $\prod_{i \in I} P(m_i) = \prod_{l \in L} P(w_l)$. Again, the case $M_\infty = \varnothing$ needs special attention. If $P(m_\infty) = 0$, then both sequences have value 0 and are equal. If $P(m_\infty) = 1$, then repeating m_∞ does not affect the value of the sequence. If $P(m_\infty) > 1$, then we have $\prod_{l \in L} P(w_l) = \infty$ due to the padding, but in this case also $\prod_{i \in I} P(m_i) = \infty$ by (†) and the equality still holds.

We can finally define the sequence $(v_j)_{j \in J} \in \mathsf{splits}_J(m)$. Notice that $|L| = |J|$ since J is infinite and both I_0 and $\omega \times \omega$ are countable. We thus have a bijection $f \colon J \to L$ and can set $v_j = w_{f(j)}$. Since f is bijective, $(v_j)_{j \in J}$ has the same product and value as $(w_l)_{l \in L}$, and thus also as $(m_i)_{i \in I}$.

We still have to prove that $Q(m) \leq R(m)$. (This does not immediately follow from the above argument, since we have to sum over all sequences, but different sequences $(m_i)_{i \in I}$ could be mapped to the same sequence $(v_j)_{j \in J}$.) We proceed by a case distinction using Lemma 21. The case $Q(m) = 0$ is trivial. If $Q(m) = 1$, then the construction of $(v_j)_{j \in J}$ witnesses $R(m) \geq 1$. The only other possibility is $Q(m) = \infty$. If there is a sequence $(m_i)_{i \in I} \in \mathsf{splits}_I(m)$ with value ∞, then by the above construction also $R(m) = \infty$. Otherwise there must be a sequence (in fact infinitely many) $(m_i)_{i \in I} \in \mathsf{splits}_I(m)$ with value $1 < s < \infty$. At least two distinct monomials must occur in $(m_i)_{i \in I}$ (otherwise the value would be 0, 1, or ∞), and, by construction, these monomials must also be contained in the sequence $(v_j)_{j \in J}$. It follows that there are infinitely many pairwise different permutations of $(v_j)_{j \in J}$, and since all of these sequences occur in the summation we have $R(m) = \infty$. ◀

Most of the other requirements for ∞-semirings follow by applying the properties of infinitary operations in $\mathbb{N}^\infty$ to coefficients and exponents.

▶ **Theorem 23.** $\mathbb{N}^\infty[\![X^\infty]\!]$ *is a strongly distributive* ∞*-semiring.*

Proof. It follows directly from the definition that addition and multiplication in $\mathbb{N}^\infty[\![X^\infty]\!]$ are compatible with finite operations and respect neutral elements, and we have already considered infinite powers in Lemma 22. It then suffices to prove partition invariance and strong distributivity, as these imply all remaining properties.

Partition invariance of addition follows immediately from the respective property of $\mathbb{N}^\infty$, as addition is defined by adding coefficients. For multiplication, fix a partition $(I_j)_{j \in J}$ of I. Using strong distributivity of $\mathbb{N}^\infty$, it remains to prove for each monomial m:

$$\begin{aligned}
\Big(\prod_{j \in J} \prod_{i \in I_j} P_i\Big)(m) &= \sum_{(m_j)_j \in \mathsf{splits}_J(m)} \prod_{j \in J} \sum_{(v_i)_i \in \mathsf{splits}_{I_j}(m_j)} \prod_{i \in I_j} P_i(v_i) \\
&= \sum_{(m_j)_j \in \mathsf{splits}_J(m)} \sum_{f \in F} \prod_{j \in J} \prod_{i \in I_j} P_i(f(j,i)) \\
&\overset{!}{=} \sum_{(u_i)_i \in \mathsf{splits}_I(m)} \prod_{i \in I} P_i(u_i).
\end{aligned}$$

Here, F is the set of choice functions that choose $(v_i)_i \in \mathsf{splits}_{I_j}(m_j)$ for each j. To simplify the presentation, we let $f(j,i) = v_i$ for the chosen sequence (i.e., we include the index i as argument).

We prove both directions of the last equality. First let $(u_i)_i \in \mathsf{splits}_I(m)$. Set $m_j = \prod_{i \in I_j} u_i$. Then $(m_j)_j \in \mathsf{splits}_J(m)$ by comparing exponents and using partition invariance of $\mathbb{N}^\infty$. For $i \in I_j$, we define $f(j,i) = u_i$. Since each i occurs in exactly one I_j, we have $\prod_{j \in J} \prod_{i \in I_j} P_i(f(j,i)) = \prod_{i \in I} P_i(u_i)$ by partition invariance of $\mathbb{N}^\infty$. Notice that our construction $(u_i)_i \mapsto ((m_j)_j, f)$ is injective, so direction $\geq$ holds (by partition invariance of addition in $\mathbb{N}^\infty$).

For the other direction, let $(m_j)_j \in \mathsf{splits}_J(m)$ and $f \in F$. For each $i \in I$, pick the unique j with $i \in I_j$ and set $u_i = f(j,i)$. Then (by partition invariance of $\mathbb{N}^\infty$ in each exponent):

$$\prod_{i \in I} u_i = \prod_{j \in J} \prod_{i \in I_j} u_i = \prod_{j \in J} \prod_{i \in I_j} f(j,i) = \prod_{j \in J} m_j = m,$$

so $(u_i)_i \in \mathsf{splits}_I(m)$. Applying the same argument to the coefficients yields

$$\prod_{i \in I} P_i(u_i) = \prod_{j \in J} \prod_{i \in I_j} P_i(u_i) = \prod_{j \in J} \prod_{i \in I_j} P_i(f(j,i)).$$

Again, the mapping $((m_j)_j, f) \mapsto (u_i)_i$ we construct is injective, so direction $\leq$ holds as well.

To prove strong distributivity, let $(I_j)_{j \in J}$ be a partition of I and F be the set of choice functions f with $f(j) \in I_j$. Strong distributivity then follows from strong distributivity and partition invariance of $\mathbb{N}^\infty$:

$$\begin{aligned}
\Big(\prod_{j \in J} \sum_{i \in I_j} P_i\Big)(m) &= \sum_{(m_j)_j \in \mathsf{splits}_J(m)} \prod_{j \in J} \sum_{i \in I_j} P_i(m_j) \\
&= \sum_{(m_j)_j \in \mathsf{splits}_J(m)} \sum_{f \in F} \prod_{j \in J} P_{f(j)}(m_j) \\
&= \sum_{f \in F} \sum_{(m_j)_j \in \mathsf{splits}_J(m)} \prod_{j \in J} P_{f(j)}(m_j) \\
&= \Big(\sum_{f \in F} \prod_{j \in J} P_{f(j)}\Big)(m).
\end{aligned}$$

◀

We now establish a kind of universal property of $\mathbb{N}^\infty[\![X^\infty]\!]$. This shows, in particular, that $\mathbb{N}^\infty[\![X^\infty]\!]$ is the free strongly distributive ∞-semiring.

▶ **Theorem 24** (κ-universality)**.** *Let $\mathcal{S}$ be a κ-distributive ∞-semiring. Every mapping $h\colon X \to \mathcal{S}$ extends uniquely to a semiring κ-homomorphism $h\colon \mathbb{N}^\infty[\![X^\infty]\!] \to \mathcal{S}$.*

Proof. For every element $s \in \mathcal{S}$ there exist unique elements $\infty \cdot s = \sum_{i \in I} s$ and $s^\infty = \prod_{i \in I} s$ for all infinite index sets I. Thus, $n \cdot s$ and s^n are well-defined for all $s \in \mathcal{S}$ and $n \in \mathbb{N}^\infty$. We first lift h to monomials m by setting $h(m) := \prod_{x \in X} h(x)^{m(x)}$. By partition invariance, h commutes with (finite and infinitary) products of monomials:

$$h(\prod_{i \in I} m_i) = \prod_{x \in X} h(x)^{\sum_{i \in I} m_i(x)} = \prod_{x \in X} \prod_{i \in I} h(x)^{m_i(x)} = \prod_{i \in I} \prod_{x \in X} h(x)^{m_i(x)} = \prod_{i \in I} h(m_i).$$

We write power series $P \in \mathbb{N}^\infty[\![X^\infty]\!]$ as $P = \sum_{m \in M}(P(m) \cdot m)$. Then h is uniquely defined by $h(P) := \sum_{m \in M}(P(m) \cdot h(m))$. We need to show that h commutes with the (finite and infinitary) semiring operations. Since infinitary operations are compatible with the finite ones, it suffices to prove that h commutes with the infinitary sum and product. This is easy for summation (over index sets of arbitrary cardinality) due to partition invariance:

$$h\Big(\sum_{i\in I} P_i\Big) = \sum_{m\in M}\Big(\sum_{i\in I} P_i(m)\Big)\cdot h(m) = \sum_{m\in M}\sum_{i\in I}(P_i(m)\cdot h(m)) = \sum_{i\in I} h(P_i).$$

For products, we use partition invariance (pi) and strong distributivity (sd) for the cardinality of I. Let F be the set of (unrestricted) functions $f\colon I\to M$. Then,

$$\begin{aligned}\prod_{i\in I} h(P_i) &= \prod_{i\in I}\sum_{m\in M}(P_i(m)\cdot h(m)) \overset{\text{(sd)}}{=} \sum_{f\in F}\prod_{i\in I} P_i(f(i))\cdot h(f(i))\\ &\overset{\text{(pi)}}{=} \sum_{m\in M}\sum_{(m_i)_i\in\mathsf{splits}(m)}\Big(\prod_{i\in I}(P_i(m_i)\cdot h(m_i))\Big)\\ &\overset{\text{(pi)}}{=} \sum_{m\in M}\sum_{(m_i)_i\in\mathsf{splits}(m)}\Big(\prod_{i\in I} P_i(m_i)\cdot\prod_{i\in I} h(m_i)\Big)\\ &= \sum_{m\in M}\sum_{(m_i)_i\in\mathsf{splits}(m)}\Big(\Big(\prod_{i\in I} P_i(m_i)\Big)\cdot h(m)\Big)\\ &\overset{\text{(pi)}}{=} \sum_{m\in M}\Big(\sum_{(m_i)_i\in\mathsf{splits}(m)}\prod_{i\in I} P_i(m_i)\Big)\cdot h(m) = h\Big(\prod_{i\in I} P_i\Big).\end{aligned}$$ ◀

▶ Remark 25. Notice that Theorem 24 no longer holds if we drop the exponent ∞. It is not clear how the infinite power $x\cdot x\cdot x\cdots$ is then defined, but we can argue by case distinction that in any case, the universal property is violated.

- If $\prod_{i<\omega} x = 0$, then $h(\prod_{i<\omega} x) = 0 \neq 1 = \prod_{i<\omega} h(x)$ for $h(x)=1$ (say in the Viterbi semiring).
- If $P = \prod_{i<\omega} x \neq 0$, then there must a monomial m (with finite exponents) and coefficient $P(m) = n > 0$. For $h(x) = \frac{1}{2}$ into the Viterbi semiring, we then get $h(P) \geq h(n\cdot m) = n\cdot h(m) = h(m) > 0$, but also $\prod_{i<\omega} h(x) = (\frac{1}{2})^\infty = 0$, contradiction.

▶ Remark 26. Observe that $\mathbb{N}^\infty[\![X^\infty]\!]$ is not fully continuous. To see this, consider the decreasing chain $P_i = \sum_{i<j<\omega} x^j$, where the monomial x^i disappears in the i-th step and the infimum is thus 0. Then $x^\infty\cdot\bigsqcap_i P_i = x^\infty\cdot 0 = 0$, but $\bigsqcap_i(x^\infty\cdot P_i) = \bigsqcap_i \infty\cdot x^\infty = \infty\cdot x^\infty$.

Similar constructions of universal infinitary semirings are possible for smaller classes of semirings. For idempotent semirings, the appropriate semiring is $\mathbb{B}[\![X^\infty]\!]$, which is constructed in the same way as $\mathbb{N}^\infty[\![X^\infty]\!]$ but with Boolean coefficients. The above proofs can easily be adapted to show that $\mathbb{B}[\![X^\infty]\!]$ is a strongly distributive ∞-semiring and satisfies κ-universality for all idempotent κ-distributive semirings.

For absorptive semirings, the appropriate choice are generalised absorptive polynomials $\mathbb{S}^\infty(X)$, for a finite set X of indeterminates. These are known to be the freely generated absorptive, fully-continuous semirings (cf. [1]). Since fully-continuous homomorphisms are also ∞-homomorphisms (by Proposition 19), $\mathbb{S}^\infty(X)$ is universal also for all infinitary absorptive semirings (notice that we do not have to assume strong distributivity here). More recently, a version of $\mathbb{S}^\infty(X)$ with infinite indeterminate set X was studied in [11]. The resulting semiring is no longer fully continuous, but it is still κ-universal for infinitary absorptive semirings in the sense of Theorem 24 (i.e., assuming κ-distributivity of the target semiring).

Additionally requiring idempotence of multiplication leads to the class of lattice semirings. For a finite set X of indeterminates, the freely generated lattice semiring, also called PosBool(X), is finite and hence infinitary operations become trivial. For infinite X, one can consider the free completely distributive lattice (see, e.g., [10]) which, in our terminology, is universal for all strongly distributive infinitary lattice semirings.

6 Semiring Provenance for First-Order Logic

For a given finite relational vocabulary τ, we denote by $\mathrm{Lit}_n(\tau)$ the set of literals $R\bar{x}$ and $\neg R\bar{x}$ where $R \in \tau$ and $\bar{x}$ is a tuple of variables from $\{x_1, \dots, x_n\}$. The set $\mathrm{Lit}_A(\tau)$ refers to literals $R\bar{a}$ and $\neg R\bar{a}$ that are instantiated with elements from a universe A.

▶ **Definition 27** ($\mathcal{S}$-interpretation). *Given an ∞-semiring, $\mathcal{S}$, a mapping $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ is an $\mathcal{S}$-interpretation (of vocabulary τ and universe A). We say that $\mathcal{S}$ is model-defining if exactly one of the values $\pi(L)$ and $\pi(\neg L)$ is zero for any pair of complementary literals $L, \neg L \in \mathrm{Lit}_A(\tau)$.*

An $\mathcal{S}$-interpretation $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ inductively extends to valuations $\pi[\![\varphi(\bar{a})]\!]$ of instantiated first-order formulae $\varphi(\bar{x})$ in negation normal form. Equalities are interpreted by their truth value, that is $\pi[\![a = a]\!] := 1$ and $\pi[\![a = b]\!] := 0$ for $a \neq b$ (and analogously for inequalities). Based on that, the semantics of disjunction and existential quantifiers is defined via (possibly infinitary) sums, while conjunctions and universal quantifiers are interpreted as (possibly infinitary) products.

$$\pi[\![\psi \vee \vartheta]\!] := \pi[\![\psi]\!] + \pi[\![\vartheta]\!] \qquad \pi[\![\psi \wedge \vartheta]\!] := \pi[\![\psi]\!] \cdot \pi[\![\vartheta]\!]$$

$$\pi[\![\exists x \varphi(x, \bar{b})]\!] := \sum_{a \in A} \pi[\![\varphi(a, \bar{b})]\!] \qquad \pi[\![\forall x \varphi(x, \bar{b})]\!] := \prod_{a \in A} \pi[\![\varphi(a, \bar{b})]\!]$$

Negation is handled via negation normal form (denoted nnf), i.e. for every $\psi \in \mathrm{FO}$ we identify $\pi[\![\neg\psi]\!]$ with $\pi[\![\mathrm{nnf}(\neg\psi)]\!]$. This will allow us to compare valuations $\pi[\![\psi]\!]$ and $\pi[\![\neg\psi]\!]$ in a meaningful way. We now examine, which of the basic properties of first-order provenance, as listed for instance in [6] extend to the infinitary case. We start with the *fundamental property for first-order provenance* which is just the simple fact that semiring valuations are compatible with semiring homomorphisms. This obviously translates to the infinitary case, for homomorphisms that also preserve infinitary sums and products.

▶ **Proposition 28** (Fundamental Property). *Let $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ be an $\mathcal{S}$-interpretation with universe A of cardinality at most κ, and let $h\colon \mathcal{S} \to \mathcal{S}'$ be a semiring κ-homomorphism. Then, $(h \circ \pi)$ is an $\mathcal{S}'$-interpretation and $h(\pi[\![\varphi(\bar{a})]\!]) = (h \circ \pi)[\![\varphi(\bar{a})]\!]$ for all $\varphi(\bar{x}) \in \mathrm{FO}(\tau)$ and instantiations $\bar{a} \subseteq A$.*

Naturally ordered semirings are $+$-*positive*, and this trivially extends to infinite sums: $\sum_{i \in I} s_i = 0$ only if $s_i = 0$ for all $i \in I$. Recall that a semiring is *positive* if it is $+$-positive and has no divisors of 0. However, in many relevant positive semirings it my be the case that an infinite product evaluates to 0, although all its factors are positive. Simple examples are products $\prod_{i<\omega} s_i$ with $s_i \leq 1 - \varepsilon$ in the Viterbi semiring. We shall see that this may lead to the sometimes undesirable effect, that the semiring valuation of a true sentence may evaluate to 0.

▶ **Definition 29.** *We call an ∞-semiring ∞-positive if it is positive, and any infinitary product of non-zero elements is also non-zero.*

The characterisation of positive semirings by homomorphisms into the Boolean semiring extends to ∞-positivity.

▶ **Lemma 30.** *An ∞-semiring $\mathcal{S}$ is ∞-positive if, and only if, $\dagger_S\colon S \to \mathbb{B}$, defined by*

$$\dagger_S(s) = \begin{cases} \top & \text{if } s \neq 0 \\ \bot & \text{if } s = 0 \end{cases}$$

is an ∞-homomorphism.

A model-defining $\mathcal{S}$-interpretation $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ defines the unique (classical) structure $\mathfrak{A}_\pi$ with universe A, such that $\mathfrak{A}_\pi \models L$ for a literal $L \in \mathrm{Lit}_A(\tau)$ if, and only if, $\pi(L) \neq 0$. We can identify $\mathfrak{A}_\pi$ with its canonical $\mathbb{B}$-interpretation $\dagger_{\mathcal{S}} \circ \pi$ which maps the true literals to $\top$, and the false ones to $\bot$. (Notice that $\dagger_{\mathcal{S}} \circ \pi$ is well-defined for every semiring, although $\dagger_{\mathcal{S}}$ is an ∞-homomorphism, only for ∞-positive semirings.)

Here is an associated semantical notion. We call $\mathcal{S}$ *truth-preserving* (for first-order logic) if for every model-defining $\mathcal{S}$-interpretation and every sentence $\psi \in \mathrm{FO}(\tau)$ we have that $\pi[\![\psi]\!] \neq 0$ if, and only if $\mathfrak{A}_\pi \models \psi$.

▶ **Proposition 31.** *A ∞-semiring $\mathcal{S}$ is truth-preserving if, and only if, it is ∞-positive.*

Proof. If $\mathcal{S}$ is ∞-positive then $\dagger_{\mathcal{S}}$ is an ∞-homomorphism. Given any model-defining $\mathcal{S}$-interpretation π, then by the fundamental property,

$$\mathfrak{A}_\pi \models \psi \iff (\dagger_{\mathcal{S}} \circ \pi)[\![\psi]\!] = \top \iff \dagger_{\mathcal{S}}(\pi[\![\psi]\!]) = \top \iff \pi[\![\psi]\!] \neq 0.$$

Conversely, assume that $\mathcal{S}$ is an infinitary semiring that is not ∞-positive. Then there exists a non-empty finite or infinite sequence $(s_a)_{a\in A}$ such $s_a \neq 0$ for all $a \in A$, but $\prod_{a\in A} s_a = 0$. We use this to define an $\mathcal{S}$-interpretation with universe A and one unary predicate P such that $\pi(Pa) = s_a$ (and $\pi(\neg Pa) = 0$) for all $a \in A$. The model defined by π is $\mathfrak{A}_\pi = (A, P)$ with $P = A$, and clearly $\mathfrak{A}_\pi \models \forall x Px$. However, $\pi[\![\forall x Px]\!] = \prod_{a\in A} s_a = 0$, so $\mathcal{S}$ is not truth-preserving. ◀

Many interesting semiring interpretations in provenance analysis do not define a single structure but a whole class of structures (with common universe and common vocabulary). In general, we can assume that such interpretations are consistent, in the sense that valuations of complementary literals satisfy certain constraints, although they need not be as strict as those for model-defining interpretations. We examine how such constraints for literals constrain the valuations of arbitrary first-order sentences.

▶ **Proposition 32.** *Let $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ be a $\mathcal{S}$-interpretation.*

- *If for every $L \in \mathrm{Lit}_A(\tau)$ at least one of $\pi(L)$ and $\pi(\neg L)$ is 0 then for any sentence $\psi \in \mathrm{FO}$, at least one of $\pi[\![\psi]\!]$ and $\pi[\![\neg\psi]\!]$ is 0.*
- *If for every $L \in \mathrm{Lit}_A(\tau)$ we have $\pi(L) \cdot \pi(\neg L) = 0$ then for any sentence ψ we have $\pi[\![\psi]\!] \cdot \pi[\![\neg\psi]\!] = 0$.*

Proof. If ψ is not a literal, then there exists a finite or infinite collection of $(\varphi_i)_{i\in I}$ of sentences such that one of the values $\pi[\![\psi]\!]$ and $\pi[\![\neg\psi]\!]$ is the sum $\sum_{i\in I}[\![\varphi_i]\!]$, and the other is the product $\prod_{i\in I} \pi[\![\neg\varphi_i]\!]$. To prove the first claim, assume that $\pi[\![\psi]\!]$ and $\pi[\![\neg\psi]\!]$ are both non-zero. It follows that all values $\pi[\![\neg\varphi_i]\!]$ are non-zero. But by induction hypothesis, this implies that all values $\pi[\![\varphi_i]\!]$, and hence also their sum, must be 0, so we have a contradiction.

For the second claim, assume by induction hypothesis, that $\pi[\![\varphi_i]\!] \cdot \pi[\![\neg\varphi_i]\!] = 0$ for all $i \in I$. With weak distributivity, it then follows that

$$\begin{aligned}\pi[\![\psi]\!] \cdot \pi[\![\neg\psi]\!] &= \sum_{i\in I} \pi[\![\varphi_i]\!] \cdot \prod_{j\in I} \pi[\![\neg\varphi_j]\!] = \sum_{i\in I}\Big(\pi[\![\varphi_i]\!] \cdot \prod_{j\in I} \pi[\![\neg\varphi_j]\!]\Big) \\ &= \sum_{i\in I}\Big(\pi[\![\varphi_i]\!] \cdot \pi[\![\neg\varphi_i]\!] \cdot \prod_{j\in I\setminus\{i\}} \pi[\![\neg\varphi_j]\!]\Big) = 0.\end{aligned}$$ ◀

Proposition 32 holds in arbitrary ∞-semirings and supports a kind of "consistency", with the two kinds coinciding when the semiring has no divisors of 0. A related question concerns the constraint that complementary literals are not both mapped to 0, i.e. they are not both considered false under the same interpretation. We would like to conclude that this constraint as well translates to arbitrary sentences. But this is, in general, true only for ∞-positive ∞-semirings.

▶ **Proposition 33.** *Let $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ be an $\mathcal{S}$-interpretation into a ∞-positive ∞-semiring. If for every $L \in \mathrm{Lit}_A(\tau)$ we have $\pi(L) \neq 0$ or $\pi(\neg L) \neq 0$ (equivalently, $\pi(L) + \pi(\neg L) \neq 0$) then for any sentence ψ we have $\pi[\![\psi]\!] \neq 0$ or $\pi[\![\neg\psi]\!] \neq 0$ (equivalently, $\pi[\![\psi]\!] + \pi[\![\neg\psi]\!] \neq 0$).*

Proof. Towards a contradiction, suppose that $\pi[\![\psi]\!] = \pi[\![\neg\psi]\!] = 0$. As in the proof above, take sentences $(\varphi_i)_{i\in I}$ such that one of the values $\pi[\![\psi]\!]$ and $\pi[\![\neg\psi]\!]$ is the (finite or infinite) sum $\sum_{i\in I}[\![\varphi_i]\!]$, and the other is the (finite or infinite) product $\prod_{i\in I}\pi[\![\neg\varphi_i]\!]$. Since $\mathcal{S}$ is ∞-positive, it follows that $\pi[\![\neg\varphi_i]\!] = 0$ for at least one $i \in I$. By induction hypothesis, $\pi[\![\varphi_i]\!] \neq 0$, which, by $+$-positivity, contradicts the assumption that $\pi[\![\psi]\!] = 0$. ◀

The example given in the proof of Proposition 31 shows that the condition of ∞-positivity is necessary for this proposition.

7 Proof Trees

A fundamental theorem for the provenance analysis of first-order logic says that, for every semiring interpretation (over a finite domain) the valuation of a first-order sentence coincides with the sum of the valuations for its proof trees or, equivalently, the sum of the valuations of the strategies for the verifier in the associated model checking game. In game theoretic terms this has been shown in [5], and in terms of proof trees, this is presented in [6]. The question arises under which conditions this theorem generalises to semiring interpretations over infinite domains. For this purpose, we inspect the proof of the Sum-of-Proof-Trees-Theorem [6, Sect. 3.5] to see what properties of the infinitary semiring operations are needed for extending the proof to infinite domains. We first recall the relevant definitions.

An *evaluation tree* for a sentence $\psi \in \mathrm{FO}(\tau)$ on a (possibly infinite) universe A is a directed tree $\mathcal{T}$ whose nodes are labelled by formulae $\varphi(\bar{a})$, where $\varphi(\bar{x})$ is an occurrence[2] of a subformula in ψ whose free variables $\bar{x}$ are instantiated by a tuple $\bar{a}$ of elements from A, such that the following conditions hold.

- The root of $\mathcal{T}$ is ψ.
- A node $\varphi \vee \vartheta$ has one child which is labelled by either φ or ϑ.
- A node $\varphi \wedge \vartheta$ has two children labelled by φ and ϑ, respectively.
- A node $\exists y\, \varphi(\bar{a}, y)$ has one child labelled $\varphi(\bar{a}, b)$ for some $b \in A$.
- A node $\forall y\, \varphi(\bar{a}, y)$ has for each for all $b \in A$ a child labelled by $\varphi(\bar{a}, b)$ for all $b \in A$.
- The leaves of $\mathcal{T}$ are literals $L \in \mathrm{Lit}_A(\tau)$.

For any literal L, $\#_L(\mathcal{T}) \in \mathbb{N} \cup \{\infty\}$ denotes the number of occurrences of L in $\mathcal{T}$. The valuation of $\mathcal{T}$ for a semiring interpretation $\pi : \mathrm{Lit}_A(\tau) \to \mathcal{S}$ into an infinitary semiring $\mathcal{S}$ is defined as

$$\pi(\mathcal{T}) := \prod_{L\in \mathrm{Lit}_A(\tau)} \pi(L)^{\#_L(\mathcal{T})}.$$

Since $\mathcal{S}$ has unique infinite powers, there is a well-defined value $\pi(L)^\infty \in \mathcal{S}$, hence $\pi(\mathcal{T})$ is well-defined for all $\mathcal{S}$-interpretations π into ∞-semirings.

[2] Notice that we consider different occurrences of the same subformula as separate objects. In particular, a sentence $\varphi \vee \varphi$ has twice as many evaluation trees as φ.

A *proof tree* for π and $\psi \in \mathrm{FO}(\tau)$ is an evaluation tree $\mathcal{T}$ with $\pi(\mathcal{T}) \neq 0$. If π is clear from the context, we write $T(\psi)$ for the set of all proof trees for π and ψ.

▶ **Theorem 34** (Sum of Proof Trees). *Let A be domain of cardinality $< \kappa$, and let $\mathcal{S}$ be a κ-distributive ∞-semiring. For every semiring interpretation $\pi\colon \mathrm{Lit}_A(\tau) \to \mathcal{S}$ and every sentence $\psi \in \mathrm{FO}(\tau)$, we have that*

$$\pi[\![\psi]\!] = \sum_{\mathcal{T} \in T(\psi)} \pi(\mathcal{T}).$$

Proof. We proceed by induction on ψ.

- Let ψ be a literal. If $\pi(\psi) = 0$ then ψ has no proof tree, so the sum over the valuations of its proof trees is 0. Otherwise ψ has precisely one proof tree which is the literal itself. In both cases, the desired equality holds trivially.
- Let $\psi = \varphi \vee \vartheta$. A proof tree $\mathcal{T}$ for ψ has the root ψ followed by a proof tree $\mathcal{T}'$ for either φ or for ϑ; clearly $\pi(\mathcal{T}) = \pi(\mathcal{T}')$. Thus

$$\pi[\![\psi]\!] = \pi[\![\varphi]\!] + \pi[\![\vartheta]\!] = \sum_{\mathcal{T}' \in T(\varphi)} \pi(\mathcal{T}') + \sum_{\mathcal{T}' \in T(\vartheta)} \pi(\mathcal{T}') = \sum_{\mathcal{T} \in T(\psi)} \pi(\mathcal{T}).$$

- Let $\psi = \varphi \wedge \vartheta$. A proof tree $\mathcal{T}$ for ψ has the root ψ, attached to which are a proof tree $\mathcal{T}'$ for φ and a proof tree $\mathcal{T}''$ for ϑ. We can thus identify every $\mathcal{T} \in T(\psi)$ with a pair $(\mathcal{T}', \mathcal{T}'') \in T(\varphi) \times T(\vartheta)$, and since $\#_L(\mathcal{T}) = \#_L(\mathcal{T}') + \#_L(\mathcal{T}'')$ for every literal L we have that, $\pi(\mathcal{T}) = \pi(\mathcal{T}')\pi(\mathcal{T}'')$. It follows, by weak distributivity, that

$$\begin{aligned}\pi[\![\psi]\!] = \pi[\![\varphi]\!] \cdot \pi[\![\vartheta]\!] &= \sum_{\mathcal{T}' \in T(\varphi)} \pi(\mathcal{T}') \cdot \sum_{\mathcal{T}'' \in T(\vartheta)} \pi(\mathcal{T}'') \\ &= \sum_{(\mathcal{T}', \mathcal{T}'') \in T(\psi)} \pi(\mathcal{T}')\pi(\mathcal{T}'') = \sum_{\mathcal{T} \in T(\psi)} \pi(\mathcal{T}).\end{aligned}$$

- If $\psi = \exists y\, \varphi(y)$, then a proof tree $\mathcal{T}$ for ψ consists of the the root ψ, attached to which is a proof tree $\mathcal{T}_a$ for $\varphi(a)$, for some $a \in A$. Clearly $\pi(\mathcal{T}) = \pi(\mathcal{T}_a)$. It follows, by partition invariance of the infinitary sum, that

$$\pi[\![\psi]\!] = \sum_{a \in A} \pi[\![\varphi(a)]\!] = \sum_{a \in A} \sum_{\mathcal{T}_a \in T(\varphi(a))} \pi(\mathcal{T}_a) = \sum_{\mathcal{T} \in T(\psi)} \pi(\mathcal{T}).$$

- Let finally $\psi = \forall y\, \varphi(y)$. A proof tree for ψ consists of the the root ψ attached to which are proof trees $\mathcal{T}_a$ for $\varphi(a)$, for all $a \in A$. We can thus identify such a proof tree with a choice function $\mathcal{T}$ that associates with every $a \in A$ a proof tree $\mathcal{T}_a \in T(\varphi(a))$, and thus $T(\psi)$ with the set of such choice functions. Further, for every literal L, we have that $\#_L(\mathcal{T}) = \sum_{a \in A} \#_L(\mathcal{T}_a)$ and therefore $\pi(\mathcal{T}) = \prod_{a \in A} \pi(\mathcal{T}_a)$. It follows, by κ-distributivity for the index set A, that

$$\pi[\![\psi]\!] = \prod_{a \in A} \pi[\![\varphi(a)]\!] = \prod_{a \in A} \sum_{\mathcal{T} \in T(\varphi(a))} \pi(\mathcal{T}) = \sum_{\mathcal{T} \in T(\psi)} \prod_{a \in A} \pi(\mathcal{T}_a)) = \sum_{\mathcal{T} \in T(\psi)} \pi(\mathcal{T}). \qquad ◀$$

▶ **Example 35.** To see that κ-distributivity is not only used in the proof, but is indeed necessary for the the Sum-of-Proof-Trees-Theorem, we present an example of a semiring interpretation into the Viterbi semiring $\mathbb{V} = ([0,1]_{\mathbb{R}}, \max, \cdot, 0, 1)$ (a ω_1-distributive ∞-semiring whose infinitary operations are supremum and infinite product) over the uncountable domain

$\mathcal{P}^\infty(\mathbb{N})$ of infinite sets of natural numbers and the vocabulary of one binary relation E. The interpretation $\pi\colon \mathrm{Lit}_{\mathcal{P}^\infty(\mathbb{N})}(\{E\}) \to \mathbb{V}$ is defined by

$$\pi(Eab) := \begin{cases} 0 & \text{if } a \cap b = \varnothing \\ 1 - \frac{1}{2+\min(a\cap b)} & \text{otherwise.} \end{cases}$$

Let $\psi := \forall x \exists y Exy$. If the Sum-of-Proof-Trees-Theorem were true also in this case, we would have that

$$\pi[\![\psi]\!] = \sup_{\mathcal{T}\in T(\psi)} \pi(\mathcal{T}).$$

But this is not the case. On the one side, we have that

$$\pi[\![\psi]\!] = \prod_{a\in\mathcal{P}^\infty(\mathbb{N})} \sup_{b\in\mathcal{P}^\infty(\mathbb{N})} \pi(Eab) = 1.$$

Indeed, for every infinite subset $a \subseteq \mathbb{N}$ and every $n \in \mathbb{N}$ we can take $b := a \cap [n, \infty)$, and we then have that $\pi(Eab) \geq 1 - \frac{1}{2+n}$ which implies that $\sup_{b\in\mathcal{P}^\infty(\mathbb{N})} \pi(Eab) = 1$ for every a.

On the other side, the proof trees in $T(\psi)$ are in one-to one correspondence with the functions $e\colon \mathcal{P}^\infty(\mathbb{N}) \to \mathcal{P}^\infty(\mathbb{N})$ such that $a \cap e(a) \neq \varnothing$ for all a. Let $\mathcal{T}_e$ be the proof tree associated with e. From its root ψ, $\mathcal{T}_e$ branches out to the nodes $\exists y Eay$, for all $a \in \mathcal{P}^\infty(\mathbb{N})$, each of which has a unique child Eab, namely the one with $b = e(a)$. The valuation of $\mathcal{T}_e$ is

$$\pi(\mathcal{T}_e) = \prod_{a\in\mathcal{P}^\infty(\mathbb{N})} \pi(Eae(a)) = \prod_{n\in\mathbb{N}} (1 - \tfrac{1}{2+n})^{\#\{a\colon\, \min(a\cap e(a))=n\}}.$$

Since there are uncountably many $a \in \mathcal{P}^\infty(\mathbb{N})$ there exist $n \in \mathbb{N}$ such that $\min(a \cap e(a)) = n$ for infinitely (in fact uncountably) many a. Hence $\pi(T_e)$ is an infinite product in which infinitely many factors are $(1 - \frac{1}{2+n})$, hence $\pi(T_e) = 0$. Since this holds for all e, and hence all proof trees for ψ we have that

$$\sup_{T\in\mathcal{T}(\psi)} \pi(\mathcal{T}) = 0 \neq 1 = \pi[\![\psi]\!].$$

Similar examples can be constructed for the semirings $\mathbb{T}$, $\mathbb{L}$ and $\mathbb{D}$.

An interesting application of the Sum-of-Proof-Trees-Theorem concerns interpretations into semirings with dual indeterminates, as proposed in [4] and further studied in [6].

Let $X, \bar{X}$ be two disjoint finite sets of indeterminates together with a one-to-one correspondence $X \leftrightarrow \bar{X}$, and denote by $x \in X$ and $\bar{x} \in \bar{X}$ two elements that are in this correspondence. We shall use X for positive literals $R\bar{a}$ and $\bar{X}$ for negated literals $\neg R\bar{a}$. By convention, if we annotate $R\bar{a}$ with x, then $\bar{x}$ can only be used to annotate $\neg R\bar{a}$, and vice versa. We refer to x and $\bar{x}$ as *complementary variables.*

Analogous to the construction of the semiring $\mathbb{N}[X, \bar{X}]$ of dual-indeterminate polynomials in [4] we can define the semiring of dual-indeterminate power series $\mathbb{N}^\infty[\![X^\infty, \bar{X}^\infty]\!]$ as the quotient of $\mathbb{N}^\infty[\![(X \cup \bar{X})^\infty]\!]$ via the congruence induced by $x \cdot \bar{x} = 0$, or, equivalently, as the semiring of power series with indeterminates in $X \cup \bar{X}$ whose monomials do not contain complementary variables. We can multiply such power series as above, provided that we eliminate the monomials with complementary variables afterwards.

Most of the results and applications exhibited in [6] generalise to this setting. As an example, we mention the information that the Sum-of-Proof-Trees-Theorem delivers for model-compatible interpretations.

▶ **Definition 36.** *A model-compatible* $\mathbb{N}^\infty[\![X^\infty, \bar{X}^\infty]\!]$*-interpretation is a semiring interpretation* $\pi\colon \mathrm{Lit}_A(\tau) \to \mathbb{N}^\infty[\![X^\infty, \bar{X}^\infty]\!]$ *such that for each atom* $R\bar{a}$ *one of the following (mutually exclusive) three properties holds:*

1. $\pi(R\bar{a}) = x$ *and* $\pi(\neg R\bar{a}) = \bar{x}$ *for some* $x \in X$*, or*
2. $\pi(R\bar{a}) = 0$ *and* $\pi(\neg R\bar{a}) = 1$*, or*
3. $\pi(R\bar{a}) = 1$ *and* $\pi(\neg R\bar{a}) = 0$*.*

Contrary to model-defining interpretations, a model-compatible interpretation π defines, in general, not a single structure, but a class of structures compatible with π, consisting of all structures $\mathfrak{A}$ (with universe A and vocabulary τ) such only those literals $L \in \mathrm{Lit}_A(\tau)$ can be true in $\mathfrak{A}$ for which $\pi(L) \in X \cup \bar{X} \cup \{1\}$.

▶ **Corollary 37.** *Let* $\pi\colon \mathrm{Lit}_A(\tau) \to \mathbb{N}^\infty[\![X^\infty, \bar{X}^\infty]\!]$ *be model-compatible and let* ψ *be sentence in* $\mathrm{FO}(\tau)$*. Then the power series* $\pi[\![\psi]\!]$ *describes all proof trees that verify* ψ *using premises from the literals that* π *maps to indeterminates or to 1.*

Specifically, each monomial $c\, x_1^{e_1} \cdots x_k^{e_k}$ *in* $\pi[\![\psi]\!]$ *stands for* c *distinct proof trees that use* e_1 *times the literal annotated by* x_1*, ..., and* e_k *times the literal annotated by* x_k*, where* $x_1, \dots, x_k \in X \cup \bar{X}$*. In particular, when* $\pi[\![\psi]\!] = 0$ *no proof tree exists, and hence there is no model of* ψ *that is compatible with* π*.*

8 Summary and Conclusion

Up to now, semiring provenance has essentially been restricted to finite data, typically to database queries against a finite, possibly annotated, database, to first-order sentences evaluated over a finite domain, or to the strategy analysis for a (possibly infinite) game, played on a finite game graph.

In this paper we have provided foundations for semiring provenance over infinite domains. This required to expand semirings by infinitary sum and product operators, and we have investigated in detail the necessary, or at least desirable, algebraic properties that should hold for these operators. Clearly the infinitary operators must be compatible with finite sums and products. Partition invariance, a natural generalisation of associativity, turned out to be a quite strong property which also implies bijection invariance, the natural generalisation of commutativity. However, we have seen that these basic properties do not suffice to exclude "pathological" operators, and we have investigated a number of other algebraic properties, including (strong) compactness, the respect of neutral and idempotent elements, and the existence of unique powers. We have seen that compactness is a very powerful property, which implies the other ones, but since it is sometimes hard to verify, and does in fact not hold in all interesting semirings, we have decided not to impose it as a necessary requirement in our definition of infinitary semirings. Instead we work with the weaker requirements that neutral elements are respected and that there exist unique infinite powers.

Distributivity and monotonicity are the fundamental algebraic properties that govern the interplay of sums and products in (naturally ordered) semirings. The generalisation of the distributive law from finite to infinitary semirings comes in two variants, a weak one and a strong one. While weak distributivity is unproblematic in the semirings we consider, it turned out that strong distributivity is more delicate. In some important semirings it does not hold for arbitrary index sets but only for countable ones. On the other side, strong distributivity is an important property which, for instance, implies monotonicity and is also used later in the Sum-of-Proof-Trees-Theorem. We decided to omit strong distributivity in our definition of infinitary semirings, and to require only weak distributivity and monotonicity. Instead we have introduced also the variant of a κ-distributive infinitary semiring.

Based on this algebraic analysis, and on the requirements for provenance valuations on infinite structures, we thus have come to a proposal for an appropriate definition of infinitary semirings. We have also discussed the appropriate notions of homomorphisms among infinitary semirings. We have studied how finite semirings and infinite lattice semirings can be expanded to infinitary semirings, and we have seen that the extension $\mathbb{N}^\infty$ of the natural semiring $\mathbb{N}$ is a strongly distributive infinitary semiring. For the class of absorptive semirings, multiplication is decreasing and hence infinitary operations can be defined in a natural way by suprema and infima over finite subsets. Finally we have investigated the generalisation of the universal semiring of multivariate polynomials, $\mathbb{N}[X]$, to an infinitary semiring $\mathbb{N}^\infty[\![X^\infty]\!]$ of generalised power series. We have proved that $\mathbb{N}^\infty[\![X^\infty]\!]$ is indeed universal for strongly distributive infinitary semirings.

In the last two sections, we have shown that, based on these infinitary semirings, the provenance analysis for first-order logic can indeed be extended from finite structures to infinite ones, preserving the basic results of the theory. In particular, we have proved that the Sum-of-Proof-Trees-Theorem, saying that the semiring valuation of a first-order sentence coincides with the sum of the valuations of its proof trees, also holds on domains $< \kappa$, provided that the underlying infinitary semiring is κ-distributive. Further, we have briefly discussed the use of dual indeteminates (for treating negation) which leads to semirings of dual-indeterminate generalised power series. The Sum-of-Proof-Trees-Theorem, applied to a model-compatible interpretation into such a semiring, gives valuations that describe all proof trees of a sentence, with precise information, which of the tracked literals are actually used in a proof tree, and how often.

A limitation of this result is that the semirings of generalised power series have only finitely many indeterminates. This means that although we can deal with infinite structures, we can track inside of these only finitely many literals, and take the truth values of the others for granted. Over an infinite universe, a model-compatible interpretation thus defines a class of structures in which the truth values of all but finitely many literals coincide. In this way, we can thus track finite data embedded into an infinite background structure, but not the collection of all atomic facts in an infinite structure. For the Sum-of-Proof-Trees-Theorem as such, no such restriction applies, so it can be used for provenance valuations in application semirings over arbitrary infinite domains.

To overcome this limitation of $\mathbb{N}^\infty[\![X^\infty]\!]$, we would need universal provenance semirings with infinitely many variables. We have seen that for $\mathbb{N}^\infty[\![X^\infty]\!]$ itself, the extension to infinite sets X would not be consistent with our definition of infinitary semirings, but such extensions seem possible in settings of absorptive provenance semirings such as $\mathbb{S}^\infty(X)$ and $\mathrm{PosBool}(X)$, and perhaps also for Why-semirings. But this will have to be studied elsewhere.

References

1 K. Dannert, E. Grädel, M. Naaf, and V. Tannen. Semiring provenance for fixed-point logic. In C. Baier and J. Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic (CSL 2021)*, volume 183 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:22, Dagstuhl, 2021. doi:10.4230/LIPIcs.CSL.2021.17.

2 J. Foster, T. Green, and V. Tannen. Annotated XML: queries and provenance. In *Proceedings of PODS 2008*, pages 271–280, 2008. doi:10.1145/1376916.1376954.

3 B. Glavic. Data provenance. *Foundations and Trends in Databases*, 9(3-4):209–441, 2021. doi:10.1561/1900000068.

4 E. Grädel and V. Tannen. Semiring provenance for first-order model checking. arXiv:1712.01980 [cs.LO], 2017. arXiv:1712.01980.

5 E. Grädel and V. Tannen. Provenance analysis for logic and games. *Moscow Journal of Combinatorics and Number Theory*, 9(3):203–228, 2020. Preprint available at `https://arxiv.org/abs/1907.08470`. `doi:10.2140/moscow.2020.9.203`.

6 E. Grädel and V. Tannen. Provenance analysis and semiring semantics for first-order logic. submitted for publication, 2024.

7 T. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *Principles of Database Systems PODS*, pages 31–40. ACM, 2007. `doi:10.1145/1265530.1265535`.

8 T. Green and V. Tannen. The semiring framework for database provenance. In *Proceedings of PODS*, pages 93–99. ACM, 2017. `doi:10.1145/3034786.3056125`.

9 G. Markowsky. Chain-complete posets and directed sets with applications. *Algebra universalis*, 6(1):53–68, 1976. `doi:10.1007/BF02485815`.

10 G. Markowsky. Free completely distributive lattices. *Proceedings of the American Mathematical Society*, 74(2):227–228, 1979. `doi:10.1090/S0002-9939-1979-0524290-9`.

11 M. Naaf. *Logic, Semirings, and Fixed Points.* PhD thesis, RWTH Aachen University, 2024. Forthcoming.

Annotation and More Annotation: Some Problems Posed by (and to) Val Tannen

Peter Buneman
University of Edinburgh, UK

Stijn Vansummeren
UHasselt, Data Science Institute, Belgium

Abstract

Among the many research accomplishments of Val Tannen, his work on provenance and semirings is probably the most widely known. In this paper, we discuss questions that arise when applying this general framework to the setting of curated databases, and in particular the setting where we can have multiple annotations on the same data, as well as annotations on annotations.

2012 ACM Subject Classification Information systems → Database design and models; Theory of computation → Database theory; Theory of computation → Data provenance

Keywords and phrases Annotation, provenance, semiring, curated data

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.4

Funding *Stijn Vansummeren*: Supported by the Bijzonder Onderzoeksfonds (BOF) of Hasselt University (Belgium) under Grant No. BOF20ZAP02.

Acknowledgements Val Tannen of course!

1 Background

By rights, Val Tannen should be a co-author of this short discussion because most of the ideas arose from ongoing conversations with him about the need to put several annotations on the same structure. There are various reasons for wanting to do this, the most important being that it is common in practice to find structures with multiple annotations, and the distinction between annotation and data is not always clear.

What we want to do in this paper is first to compare provenance and annotation: provenance being something that describes, or is intrinsic to, the formation of data; annotation being something that is superimposed on data after its formation. Second, to look at the commonplace practice of having multiple annotations on a structure. How do we represent annotations on annotations, and how well does this fit with the elegant theory of semirings [8] for which Val Tannen is responsible?

Provenance and annotation have been studied together by the database community for 25 years or more [2,6,16,17]. They are obviously connected: provenance is a form of annotation, and provenance may tell us how annotations should propagate through queries. Green, Karvounarakis and Tannen's original work [8] on semirings referred to them as "provenance semirings", other researchers [10,12] have used the term "annotation semirings". What we claim is that provenance and annotation are also fundamentally different and it is worth examining these differences.

1.1 Provenance

While there is an unending sequence of attempts to define, characterize, formalize and standardize provenance, all of them agree that provenance concerns the properties of the process by which something has been formed and used. Provenance of any kind is hence an account of the history of an object. As such we do not expect to modify it. Indeed, there are efforts, both in databases [14] , and more generally [15], to ensure that one does not "rewrite history". One might argue that, ideally, everything should carry its provenance, and that provenance – whatever definition one chooses – is an unalterable, intrinsic, property of an object. This does not mean that there has to be a unique model of provenance; but if one has two definitions of provenance there has to be a notion of consistency and a method to combine the two. For instance, in [9] there is a hierarchy of semirings that shows how one semiring may be "more informative" with respect to provenance than another.

1.2 Annotation

Whereas the database community started to worry seriously about provenance and annotation at the start of the millennium, the practice of annotating data was widespread in curated databases at least ten years earlier. In stark contrast to provenance, annotations are regarded as being added to, or superimposed upon, *existing* data. For instance, the widely used Swissprot/Uniprot [1] database is regarded as a *secondary* database built on top of underlying sequence data (the *primary data*). Similarly, geospatial databases may be regarded as annotations on a terrestrial coordinate system.

Informally, annotation has the following properties: annotations are placed on data *after* it has been created; annotations do not "influence" the data they annotate; and if the underlying data changes, annotations may become invalid. Of course, without a proper representation of the notions of time, influence or update in a database, we cannot expect to formulate these conditions. But, in practice they are understood and follow naturally from keeping annotated data in a separate database together with a copy or view of the underlying primary data.

The curators and users of curated databases typically know which parts of the database are imported from the primary data and which fields are added as annotation. The distinction between annotation and the primary data is hence conveyed by some elementary form of provenance although this provenance information is not necessarily explicitly represented in the database.

2 Provenance and Simple Annotation

As a practical illustration of provenance and annotation in a curated database, Table 1 shows the relational schema of the object table of the GtoPdb [13] database. This is a base table and several other tables, all of which describe various substances, implicitly inherit and extend this schema. In this table, the columns object_id, name, abbreviation, and systematic_name form the primary data. The columns last_modified and old_object_id record some rudimentary provenance information. All other columns are annotation, either commenting on the primary data, or (like in_gctp) recording whether the primary data should be exported in a particular view on the database.

What is remarkable about this schema is first that most of it is annotation and second, as we shall see, that most of the annotation fields can usefully be given a semiring structure. It is also interesting that last_modified, a field that we have associated with provenance also has a simple semiring structure.

As we noted above, provenance appears to be a form of annotation. This is in particular true for the last_modified and old_object_id columns in Table 1. Conversely, if as in [2, 9], one wants to study the propagation of annotations through queries, one should understand provenance. If, for example, we want to know whether a tuple in the result of a query should

Table 1 Schema of the GtoPdb [13] object table.

Column	Type
object_id	integer
name	varchar(1000)
abbreviation	varchar(100)
systematic_name	varchar(100)
old_object_id	integer
last_modified	date
comments	text
structural_info_comments	text
annotation_status	integer
only_iuphar	boolean
grac_comments	text
only_grac	boolean
no_contributor_list	boolean
quaternary_structure_comments	text
in_cgtp	boolean
in_gtip	boolean
gtip_comment	text
in_gtmp	boolean
gtmp_comment	text
cite_id	varchar(20)

be included in a view specified by a field in_gctp, then we should be able to use provenance to determine the value of that field. In fact one can use semirings and the semiring semantics of relational algebra to describe the propagation of such fields. If there is only a single annotation column C, and if the values in this column can be ascribed a semiring structure $\mathcal{K}$, such as is for example the case for the in_cgtp column in Table 1 where $\mathcal{K}$ is the Boolean semiring, then we can simply propagate the annotation C by means of the semiring semantics introduced in [8].

▶ Remark. Before continuing our discussion, it is worth noting that we have to be careful in choosing the semiring $\mathcal{K}$ we intend to work in. Indeed, the formal definition of $\mathcal{K}$-relations proposed in [8] interprets all tuples that are annotated with the semiring's additive unit 0 as being absent from the database, i.e., *non-existent.* This need not be the semantics that one wants. For example, in GtoPdb database and assuming that in_cgtp is the only annotation attribute, a value of 0 in in_cgtp does not mean that the tuple is non-existent or should be deleted. Rather, it means that the tuple should not be exported to the gctp view. This difference can be resolved by moving to a semiring other than the Boolean semiring. This alternate semiring has pairs (b_1, b_2) of booleans as elements; where $b_1 = 0$ implies that $b_2 = 0$ and where $b_1 = 0$ indicates that the tuple is really not present. Otherwise, $b_1 = 1$ indicates tuple presence and b_2 indicates whether the tuple is exported to in_gctp. The semiring operations are defined pointwise in the obvious manner. For reasons of parsimony we will ignore this issue and continue to work with the Boolean semirings for fields such as in_gctp.

The GtoPdb object table (Table 1) illustrates the common practice in curated databases of having multiple annotations; and this leads to the following question, central to this paper.

How does one collectively *propagate this multitude of annotations? In particular, can we always cast this as a form of semiring provenance and, if so, what is the semiring structure required? Moreover, how may we implement such propagation?*

$$I_1 = \begin{array}{ccc||ccc} A & B & \dots & X & Y & Z \\ \hline a_1 & b_1 & \dots & 1 & 1 & 0 \\ a_2 & b_2 & \dots & 0 & 1 & 0 \\ a_3 & b_3 & \dots & 0 & 0 & 0 \\ a_4 & b_4 & \dots & 1 & 1 & 1 \end{array} \qquad I_2 = \begin{array}{ccc||c} A & B & \dots & \mathcal{A} \\ \hline a_1 & b_1 & \dots & \{X, Y\} \\ a_2 & b_2 & \dots & \{Y\} \\ a_3 & b_3 & \dots & \emptyset \\ a_4 & b_4 & \dots & \{X, Y, Z\} \end{array}$$

$$I_3 = \begin{array}{ccc||c} A & B & \dots & \mathcal{A}' \\ \hline a_1 & b_1 & \dots & X \\ a_1 & b_1 & \dots & Y \\ a_2 & b_2 & \dots & Y \\ a_4 & b_4 & \dots & X \\ a_4 & b_4 & \dots & Y \\ a_4 & b_4 & \dots & Z \end{array} \quad \begin{array}{l} \text{together with the base instance } I \\ \text{and } \pi_{AB\dots} I_3 \subseteq I \end{array}$$

Figure 1 Three ways of annotating with sets.

To give some insights into this question, let us restrict our attention, at least for the purpose of this section, to the setting where there are multiple annotations, but all of these annotations are of the same form. We will return to distinct forms in Section 3.

To start our discussion, we remark that the GtoPdb object table exhibits the following two forms of annotation that commonly arise in curated databases. The first of these we shall call *believers*. This is a catch-all term we will use for people or agents that "approve" of a tuple. For example it is common for curators of a database to review (and possibly correct) parts of a database. In another scenario a database may "export" a number of smaller databases or views, and tuples may be tagged with the set of views that they should be included in. In particular, the in_gctp, in_gtip, and in_gtmp columns in Table 1 represent such tags. Note that while may view a single believer annotation as single Boolean value, the set of multiple believer annotations taken together is a *set-valued* annotation.

A second and ubiquitous form of annotation are *comments*. Again, we may view a single comment as a single string. Multiple comments are commonplace, however, and in practice they are represented by concatenating strings in such a way that the reader can separate them. Hence, a single comment annotation typically encodes a *set* of comment values. Columns like structural_info_comments in Table 1 are clearly *comments* annotations.

Consider the setting where there is only one of these forms of annotation – believers or comments – on the primary data. Figure 1 shows three possible methods of representing such annotated tables. The first, which applies only to believers, adds a new Boolean column for each curator or view to the primary database table. This method is only practical if we may assume that the set of possible curators or views is known and small. If this is not the case, and if one has the luxury of working systems that support nested tables [4] one can add a column that simply contains the set of curators (respectively, views or comments) as shown by I_2. Otherwise (and also common practice) one may add an auxiliary table – with the appropriate foreign key constraint – that specifies a binary relation between the primary data and the believers respectively comments. This third method I_3 is often used in practice when curators or contributors details are also kept in the database.

The first two representation methods I_1 and I_2 naturally lend them to a semiring-based semantics for carrying the annotations through queries. For both believers and comments these semirings are semirings of sets, straightforward but different. In the case of believers and representation method I_1, the added Boolean columns can be viewed as boolean vectors of a fixed dimension, which clearly form a semiring. If as in I_2 we are using a set-valued column then we want the lattice of subsets of believers as the appropriate semiring. For example, if tuples r, t_1 and t_2 are respectively believed by sets R, T_1 and T_2 then the set of people who believe in the tuple with provenance polynomial $r_1(t_1 + t_2)$ is $R \cap (T_1 \cup T_2)$: we are using the semiring of subsets of some set – a distributive lattice.

On the other hand, if R, T_1 and T_2 are sets of comments, we would expect to see the tuple with polynomial $r_1(t_1 + t_2)$ annotated with $R \cup T_1 \cup T_2$. Here we are using a commutative idempotent monoid – a degenerate semiring in which the two binary operations coincide and the empty set is the multiplicative unit. This is also the semiring used in lineage [6].

By contrast, representation method I_3, while ubiquitous in practice, does not immediately seem to lend itself to a semiring-based semantics. The reason is that the values in the annotation column $\mathcal{A}'$ in the auxiliary table (being the individual believers or comments) does not necessarily have a natural semiring interpretation. For: which individual believer corresponds to the product or sum of two individual believers?

We find it interesting to remark, however, that we may instead view representation method I_3 as a means to *implement* the annotation propagation semantics of method I_2 under the above-described distributive lattice and lineage semirings. Whereas a straightforward implementation of I_2 requires systems that support nested tables (as the annotation column contains a set), method I_3 is able to implement this using ordinary flat relational database management systems. In particular, when we view I_2 as a nested relation then I_3 together with the base instance I is a particular form of a so-called *shredded representation* of I_2 [5,7,11]. It is possible to simulate full nested relational algebra (NRA) by means of ordinary flat relational algebra (RA) on shredded representations [5,7], something that also Val has worked on [11]. Hence, we may indeed view I_3 and particular relational algebra queries on I_3, as a way to simulate semiring-based querying on on I_2.

In full generality, however, shredding requires that RA is endowed with the ability to invent new identifiers. This is required to simulate the creation of new (deeply) nested sets. In the particular case of annotation-propagation that we are interested in here and the specific case where the only NRA operations that we want to do on set-based attributes in I_2 are intersection and union (our semiring operations), it seems that one does not require the ability to invent new identifiers. Indeed, because there is a functional dependency from the non-annotation attributes in I_2 to the set-based annotation attribute $\mathcal{A}$, we may always "identify" a set (including newly created ones) simply by means of the tuple of non-annotation attributes. Moreover, if we only intend to propagate the annotation on I_2 through *positive* relational algebra queries, then it seems that the simulating shredded queries on I_2 will also be positive. A full verification of these conjectures has not appeared in the literature, however, to the best of our knowledge.

A connected interesting and possibly open question is the following: even if from a theoretical point of view we can always "identify" nested sets by means of the tuple of non-annotated values in method I_3, we almost certainly do not want to do this in practice because such tuples can be very wide. Indeed, in the simulating shredded queries we will often have to join using the identifier columns as join fields, and thus the fewer join fields the better. Hence the question: when is it possible to use only a subset of the non-attribute columns as set-identifier keys in the shredded representation? How does this work if we have primary key information on the primary data?

3 Multiple annotations

Beyond illustrating some distinctions between annotation and provenance, the GtoPdb example shows that multiple annotations on a tuple are commonplace. Moreover, where semirings may be associated with annotations, different semirings serve different purposes.

That we have so many annotations on a single tuple in the GtoPdb example is partly due to database design. Take, for example, the view specified by in_gtmp=true. Since object_id is a key one could place the gtmp_comment field in another table with key object_id and the appropriate key inclusion. Multiple uses of this transformation would be confusing to say the least, and the database designers rightly kept all this information in the same place.

If we do keep this information in the same place, we have to allow that two annotations may differ on what tuples are mapped to a semiring zero, so that we have to allow the explicit appearance of zero in an annotation column. As remarked in Section 2, in the familiar representation of a singly-annotated relation as a $\mathcal{K}$-relation, the tuples annotated with a zero are not present.

In Section 1.2 we made the informal observation that if we changed the underlying data then an annotation could become invalid. In the case of a tuple annotation, upon what parts of a tuple does an annotation depend? In particular does one annotation depend on another? In Table 1 it is unlikely that a change to abbreviation could alter the validity of in_gtmp. However the gtmp_comment field is only going to be present if in_gtmp is true. We can informally define the *scope* of an annotation as that part of a tuple (both non-annotation columns and annotation columns) it annotates and say that two annotations are *independent* if neither is in the scope of the other.

For independent annotations there is an obvious method of treating them as a single annotation. For example, if we have a semiring $\mathcal{K}_1$ for believers and $\mathcal{K}_2$ for comments then there is an obvious product semiring defined on $\mathcal{K}_1 \times \mathcal{K}_2$ with all operations defined pointwise. We have seen this before: in the column-based representation of I_1 in figure 1, the annotation columns X and Y taken together can be regarded as the product of two Boolean-valued annotations.

A more interesting problem arises when the annotations are *not* independent. That is when one annotation depends on, or is within the scope of, another. This is also common in curated databases, and happens with both believer and comments annotations. For example, it is possible for one curator to verify or check the work of another, so "A verifies B's verification of ..." presupposes that B verified something. It can also happen when one view is a sub-view of another, and it is possible to provide comments on top of comments, or on top of verifications.[1]

For such correlated annotations, it is not immediate how we may view them as semiring elements. In fact, it is not *a priori* clear how representation methods I_1 and I_2 of Figure 1 extend to this setting. By contrast, for method I_3, the shredded representation of the annotation, the extension is straightforward: starting with schema R we want to add two annotation attributes $\mathcal{A}$ and $\mathcal{A}'$. We construct the instances I, $I_{\mathcal{A}}$ and $I_{\mathcal{A}\mathcal{A}'}$ over the schemas R, $R \cup \{\mathcal{A}\}$ and $R \cup \{\mathcal{A}, \mathcal{A}'\}$ that satisfy the inclusions $\pi_R I_{\mathcal{A}} \subseteq I$ and $\pi_{R\cup\{\mathcal{A}'\}} I_{\mathcal{A}\mathcal{A}'} \subseteq I_{\mathcal{A}}$. The following questions now arise: can we still view this as a shredded representation of generalized version of I_2? If so, can we view the annotation attribute of generalized I_2 as having a semiring structure? In discussions that we have had with Val on this topic, it seems that a semiring structure exists, but that the elements in the domain of this semiring must be themselves $\mathcal{K}$-relations. Full answers to these questions are still open.

[1] Somewhat confusingly "A believes that B believes ..." does not imply that "B believes ..." so we really cannot regard this as an annotation on an annotation.

The comment semiring can be treated similarly. The generalization of I_3 described above works equally well for comments on comments except that we are using the degenerate semiring with only union. What is interesting is that it is common to build up chains of comments of indefinite length. This can also be easily represented in the semiring framework. If C is the domain of comments and $C*$ is the set of sequences over C then the monoid we need is the prefix-closed subsets of $C*$, which is closed under union.

4 Closing and further questions

The foregoing account of annotation in curated databases is not just a theoretical exercise. As we have already mentioned, the base "object" table in GtoPdb [13] has six Boolean fields for views and six comment fields each of which pertains to one of those fields. Tens of other tables inherit from this table; moreover many other tables in the database have a comment field. Understanding which comments are relevant to a tuple in the output of a query is a non-trivial task in practice. Another interesting field that can be regarded as an annotation, but also part of the provenance is a "time of last modification" – which may be represented in some semiring.

As our discussion illustrates, it is not always clear how to cast the propagation of annotation as an application of querying under the semiring semantics. In particular, ensuring that multiple annotations can be suitably propagated seems to require us to form semirings of increasingly rich structure: semirings of sets, product semirings, and (in Section 3), semirings of $\mathcal{K}$-relations. Further research is required to complete this picture of multiple annotation propagation and the induced semiring structure.

We close our discussion with two further topics.

We introduced the notion of the scope – the set of attributes in a tuple that is subject to some annotation. Do the rules for combining annotations still hold? For example, if a projection loses some or all the attributes in the scope of an annotation, do we want to keep that annotation?

We also defined scope informally in terms of update to a tuple. But we have no model for update, and this brings up the question of how annotation behaves under update. Provenance and update have been studied in [3], but does this tell us anything about annotation; and is there a larger picture that includes semirings?

Finally, although we have suggested that the scope of an annotation could be part of a tuple, why could it not be even broader. Could we annotate data values, tables, columns, as well as arbitrary views?

References

1 Amos Bairoch and Brigitte Boeckmann. The swiss-prot protein sequence data bank. *Nucleic acids research*, 19(Suppl):2247, 1991.

2 Deepavali Bhagwat, Laura Chiticariu, Wang-Chiew Tan, and Gaurav Vijayvargiya. An annotation management system for relational databases. *The VLDB Journal*, 14:373–396, 2005.

3 Peter Buneman, James Cheney, and Stijn Vansummeren. On the expressiveness of implicit provenance in query and update languages. *ACM Transactions on Database Systems (TODS)*, 33(4):1–47, 2008.

4 Peter Buneman, Shamim A. Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theor. Comput. Sci.*, 149(1):3–48, 1995. doi:10.1016/0304-3975(95)00024-Q.

5 James Cheney, Sam Lindley, and Philip Wadler. Query shredding: efficient relational evaluation of queries over nested multisets. In Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1027–1038. ACM, 2014. doi:10.1145/2588555.2612186.

6 Yingwei Cui and Jennifer Widom. Practical lineage tracing in data warehouses. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 367–378. IEEE, 2000.

7 Jan Van den Bussche. Simulation of the nested relational algebra by the flat relational algebra, with an application to the complexity of evaluating powerset algebra expressions. *Theor. Comput. Sci.*, 254(1-2):363–377, 2001. doi:10.1016/S0304-3975(99)00301-1.

8 Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

9 Todd J Green and Val Tannen. The semiring framework for database provenance. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 93–99, 2017.

10 Grigoris Karvounarakis and Todd J Green. Semiring-annotated data: queries and provenance? *ACM SIGMOD Record*, 41(3):5–14, 2012.

11 Christoph Koch, Daniel Lupei, and Val Tannen. Incremental view maintenance for collection programming. In Tova Milo and Wang-Chiew Tan, editors, *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 75–90. ACM, 2016. doi:10.1145/2902251.2902286.

12 Egor V Kostylev, Juan L Reutter, and András Z Salamon. Classification of annotation semirings over containment of conjunctive queries. *ACM Transactions on Database Systems (TODS)*, 39(1):1–39, 2014.

13 Adam J Pawson, Joanna L Sharman, Helen E Benson, Elena Faccenda, Stephen PH Alexander, O Peter Buneman, Anthony P Davenport, John C McGrath, John A Peters, Christopher Southan, et al. The iuphar/bps guide to pharmacology: an expert-driven knowledgebase of drug targets and their ligands. *Nucleic acids research*, 42(D1):D1098–D1106, 2014.

14 Pingcheng Ruan, Gang Chen, Tien Tuan Anh Dinh, Qian Lin, Beng Chin Ooi, and Meihui Zhang. Fine-grained, secure and efficient data provenance on blockchain systems. *Proceedings of the VLDB Endowment*, 12(9):975–988, 2019.

15 Deepak K Tosh, Sachin Shetty, Xueping Liang, Charles Kamhoua, and Laurent Njilla. Consensus protocols for blockchain-based data provenance: Challenges and opportunities. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 469–474. IEEE, 2017.

16 Y. Richard Wang and Stuart E. Madnick. A polygen model for heterogeneous database systems: The source tagging perspective. In *Proceedings of the 16th International Conference on Very Large Data Bases*, VLDB '90, pages 519–538, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc.

17 Allison Woodruff and Michael Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Proceedings 13th International Conference on Data Engineering*, pages 91–102. IEEE, 1997.

Chasing Parallelism in Aggregating Graph Queries

Alin Deutsch ✉
University of California, San Diego, CA, USA

To Val Tannen, my Doktorvater and role model

Abstract

In practice, one frequently encounters queries that extract tabular results from graph databases by employing grouping and aggregation. This paper introduces a technique for rewriting the group-by list of graph queries in order to increase aggregation parallelism in graph engines that conform to a modern instantiation of the Bulk-Synchronous-Parallel computation model.

2012 ACM Subject Classification Information systems → Graph-based database models; Information systems → Query optimization

Keywords and phrases Graph Databases, Grouping and Aggregation, Parallel Graph Computation Models, Rewriting, Constraint-based Minimization

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.5

1 Introduction

It is a long-standing tradition in both database research and practice to conceptualize data as a graph, in which vertices model the entities of the application domain and edges model their relationships. This tradition preceded the advent of the relational model, starting with the *network* database model; it co-existed with the relational model from its inception, in form of the *Entity/Relationship* model; it matured into dedicated database systems conforming to the *semistructured* model and its specializations as XML and JSON [1]; it was recently rejuvenated in form of the *property graph* data model by strong commercial demand from the industrial sector; and it culminated in the just-released GQL standard [11], the first ISO/ANSI query language standard since SQL.

An invariant throughout most of this evolution has been the need for queries that cross models by extracting data from the underlying graph and presenting it in tabular form. Research on the property graph model and this class of queries is not simply a revisit of the plethora of results developed for the semistructured model. Two recent developments raise novel research challenges:

(i) Recent industrial demand for expressing graph analytic tasks requires the output tables to be the result of grouping and aggregation, as inspired by SQL. This has led to incorporating aggregation primitives as first-class citizens into graph query languages, as opposed to treating them as an afterthought in the era of semistructured data research.

(ii) New query evaluation strategies are called for due to the recent development of novel parallel computing paradigms motivated by today's distributed and cloud infrastructures.

This paper focuses on maximizing the degree of parallelism for the aggregation task when run in a graph engine that conforms to a commercially implemented parallel computation paradigm called *Edge-Map/Vertex-Reduce (EMVR)*. The solution involves exploiting integrity constraints to rewrite the group-by list of queries in ways that expose opportunities to distribute the computation over independently working processors.

EMVR is an instantiation of Valiant's *Bulk-Synchronous Parallel (BSP)* model of computation [19]. The results presented here can be ported to other currently circulating instantiations of the BSP model for graphs.

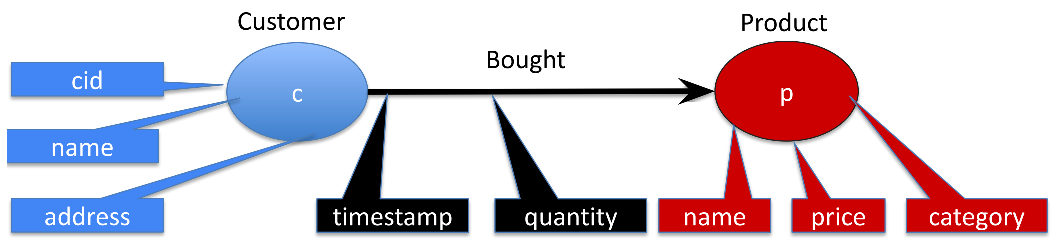


Figure 1 Modeling customers and products as vertices, and sales as edges.

The remainder of this paper is organized as follows. Section 2 describes the property graph model, and Section 3 introduces the EMVR computation paradigm. Section 4 illustrates a grouping and aggregating graph query that serves as the running example for the subsequent discussion. Section 5 illustrates a class of EMVR plans that can generically support the class of queries represented by the running example. These plans contain a parallelism-limiting bottleneck, which can be removed by exploiting knowledge of integrity constraints. Section 6 illustrates such an optimized plan for the running example. Section 7 describes our optimization approach, which is based on reducing the problem to relational minimization under integrity constraints. We conclude in Section 8.

2 The Property Graph Data Model

Per the GQL standard [11], property graphs comprise vertices and binary edges connecting them. In object-oriented fashion, vertices are uniquely determined by an internal identifier. Edges may be directed or undirected. Both vertices and edges have a set of attributes, called *properties*, which are key-value pairs with the key giving the property name. Both vertices and edges may have a type, which specifies their property names and the types of their property values. The GQL standard allows both a typed version which specifies a graph schema, and a schema-free version. Since our interest is in exploiting integrity constraints, we focus on the typed graph version here.

▶ **Example 1.** Assume we wish to model a domain revolving around customers who buy products. Individual customers are modeled as vertices of type *Customer*, with attributes *cid*, *name* and *address*. Products are modeled as vertices of type *Product*, with attributes *pname*, *price* and *category*. The fact that a customer represented by vertex c has bought a product represented by vertex p is modeled by a directed edge of type *Bought* oriented from c to p. The edge is adorned with attributes *timestamp* and *quantity* (see Figure 1). Figure 2 shows a graph modeling a history of sales. ⌟

3 The Edge-Map/Vertex-Reduce Graph Computation Model

The *Edge-Map/Vertex-Reduce (EMVR)* model of computation is an instantiation to graphs of Valiant's BSP paradigm (described below). It organizes parallel processing of the graph into *vertex functions* and *edge functions*. Each vertex/edge is thought of as an independent processor that can run a vertex/edge function in parallel with the other vertices/edges.

The computation is a sequence of steps guided by the *active vertex set*. A step can be of either vertex or edge kind. A *vertex step* maps the vertex function in parallel over all active vertices. An *edge step* maps the edge function in parallel over all edges incident to the active vertex set. Among other tasks, each step computes the next active vertex set. The next step can start only when the current step has completed (which requires a synchronization barrier).

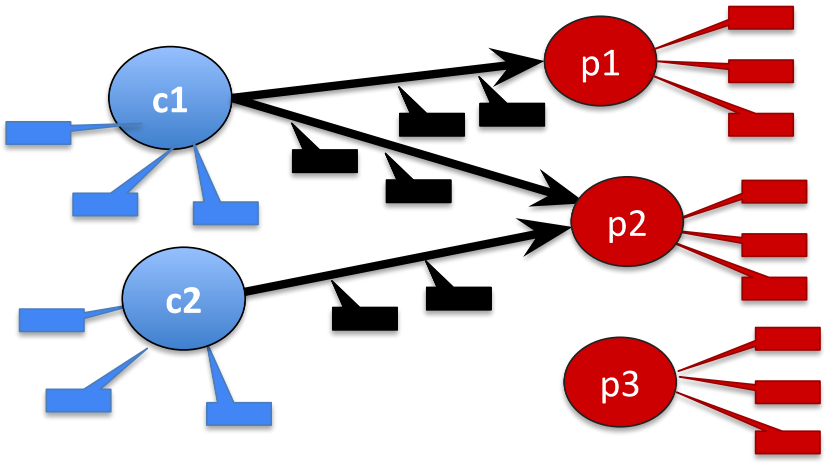


Figure 2 A graph showing multiple sales.

The *Reduce* phase of the EMVR step consists in aggregating the values generated by mapping the vertex or edge functions. In the EMVR model, aggregation is performed by *accumulators*. Accumulators are containers that store a data value, accept inputs, and aggregate these inputs into the stored data value using a binary operation $\oplus$.[1].

Since the only available EMVR implementation we are aware of is provided by the TigerGraph company, we adopt the syntax of its query language, GSQL, to denote accumulator types [7].

▶ **Example 2.** In GSQL,

$$SumAccum < float >$$

denotes the type of accumulators that sum up their floating point inputs. Each accumulator of this type has its stored value initialized to 0 at construction time. During operation, it adds each received input into the stored value using binary addition. ⌟

GSQL provides built-in accumulators that perform the standard aggregations one has come to expect of query languages (sum, average, min, max, logical or, logical and, etc.). It also supports user-defined accumulators, whose behavior is specified by providing the initial stored value and the binary operation $\oplus$.

There are two kinds of accumulators: global or vertex-attached. A *global accumulator* has a single instantiation for the entire computation. A *vertex-attached accumulator* has one instantiation for every vertex.

Accumulator implementations need to ingest inputs generated in parallel by the invocations of the vertex and edge functions. Race conditions are avoided by using an exclusive locking mechanism, which effectively serializes the inputs. The order in which inputs are incorporated into the stored value is nondeterministic. However, if the binary operation $\oplus$ is associative and commutative, the final result of the aggregation is well-defined. This is the case for all built-in accumulators (the average accumulator is implemented to maintain the sum and the count of its inputs, both of which have associative and commutative binary operations).

[1] The inspiration and theoretical underpinnings for thinking about database aggregation in this way go back to Val Tannen's work [12]

3.1 Predecessor BSP Models

Valiant's *Bulk-Synchronous Parallel (BSP)* model [19] includes three main components: a number of processors, each with its own local memory and ability to perform local computation; a communication environment that delivers messages from one processor to another; and a barrier synchronization mechanism. A BSP computation is a sequence of supersteps. A superstep comprises of a computation stage, where each processor performs a sequence of operations on local data, and a communication stage, where each processor sends a number of messages. The processors are synchronized between supersteps, i.e. they wait at the barrier until all processors have received their messages.

The *vertex-centric* computational model was introduced by Google's Pregel [15] system as an adaptation to graph data of the BSP model. In the vertex-centric model, each vertex plays the role of a processor that executes a user-defined program. Vertices communicate with each other by sending messages via outgoing edges, or directly to any other vertex whose identifier they know (e.g. discovered during computation).

Each vertex is identified by a vertex ID. It holds a state, which represents intermediate results of the computation; a list of outgoing edges; and an incoming message queue. Edges are identified by the IDs of their source and destination vertices, and they can also store state. The computation is organized in supersteps delimited by a synchronization barrier, as dictated by the BSP paradigm. During each superstep, each vertex runs the same vertex program in parallel. The vertex program is designed from the perspective of a vertex, operating on local data only: the vertex state, the received messages, and the incident edges. Based on these inputs, the vertex program can modify the vertex state, send messages to neighbors along the edges, and decide whether the vertex remains active for the next superstep. If an inactive vertex receives a message, it is reactivated. At the beginning of a computation all vertices are activated. The computation halts when all vertices are inactivated and no more messages are in transit.

3.2 Examples of BSP graph engines

A vertex-centric BSP model was first introduced in Google's Pregel [15], followed by a proliferation of open-source and commercial implementations, some running on distributed clusters, others realizing vertex communication via a shared memory. Examples include open-source implementations such as Giraph [4], GPS (Graph Processing System) [17] and Apache Spark with its Pregel API GraphX [5]. Graph engines realizing communication via a shared memory include GraphLab [13], Signal/Collect [18], and PowerGraph [8].

Surveys of the landscape can be found in [16, 20], while comparative experimental evaluations have been reported in [3, 9, 14].

More recently, TigerGraph introduced its commercial engine [7], which implements an Edge-Map/Vertex-Reduce API by exploiting thread parallelism within a server as well as distributed parallelism across computing nodes in a cluster.

4 A Grouping and Aggregating Graph Query

The following query conforms to the syntax introduced by GSQL, TigerGraph's query language. It is also close to the syntax of the recently released GQL standard [11], which is inspired by GSQL.[2]

[2] The GQL standard admits two syntactic flavors: an SQL-like one inspired by TigerGraph's GSQL, and one inspired by Neo4j's Cypher language (not shown here).

```
SELECT   c.cid, sum(b.quantity * p.price) AS revenue INTO T
FROM     (c:Customer) -[b:Bought]-> (p:Product)
WHERE    p.category = 'Toys'
GROUP BY c.cid
```

The clauses are mostly familiar from SQL.

The exception is the FROM clause, which, instead of SQL tables, specifies *graph patterns*. Round parentheses denote pattern fragments to be matched against vertices, while arrows with square parentheses are to be matched against edges (the arrow specifies the orientation for directed edges; it is omitted for undirected edges). Vertex and edge types are specified to the right of the colon delimiters, and variables to the left. In this example, the pattern matches directed edges b of type *Bought* oriented from a vertex c of type *Customer* to a vertex p of type *Product*. Each match can be thought of as a tuple whose attribute names are the three variables, and whose values are the identities of the matched vertices and edges. These tuples are collected in a bag referred to as the *binding table*. For the example sales graph in Figure 2, the binding table is

c	b	p
c1	c1 $\longrightarrow$ p1	p1
c1	c1 $\longrightarrow$ p2	p2
c2	c2 $\longrightarrow$ p2	p2

The remaining clauses manipulate the binding table analogously to SQL: the `WHERE` clause keeps only the matches whose product category is 'Toys'; the `GROUP BY` clause groups by the the customer vertex attribute *cid* (assumed here to be a key); finally, the SELECT clause aggregates the groups by summing up the product prices, to obtain the revenue per customer. The result is output into a table called `T`, as directed by the `INTO` sub-clause.

GSQL, Cypher and GQL are significantly more expressive than the example query, for instance admitting multi-hop patterns that match sequences of edges, regular expressions for specifying complex shapes of traversed paths, conjunctions of such regular expression patterns, composition of query blocks, etc. These are not discussed in this paper, which focuses on the tables produced by grouping and aggregation.

5 Supporting Tables with Accumulators

Like many real-life graph queries, our example query crosses between data models, extracting a table from the input graph data.

At first glance, this raises a challenge to supporting this class of queries in a graph engine based on the EMVR model of computation, as the latter is centered around vertices and edges and has no notion of relational tables as first-class citizens.

However, tables can be supported as syntactic sugar in the EMVR model, being implementable as accumulators (this is the case for TigerGraph's GSQL implementation). We borrow GSQL's syntax to denote the types and operations of accumulators.

For our running example query, the GSQL compiler implements table T as an accumulator of type

```
GroupByAccum<int cid, SumAccum<float> revenue>
```

which denotes accumulators that contain a set of key-value pairs, each pair corresponding to a group. In each pair, the *cid* field is the integer group key, and the *revenue* field contains the associated value, which is in turn a nested accumulator that sums up its floating point inputs (its type is denoted in GSQL as `SumAccum<float>`).

This group-by accumulator takes as input key-value pairs (k, v), aggregating them into its contents as follows. Identify the group of k and its associated *revenue* accumulator r (if no such group exists yet, create one, initializing r's contents to 0). Insert the value v into r; since r is of type `SumAccum<float>`, this will add v to the current stored value of r.

An EMVR plan for the example query is shown below.

```
construct global accumulator @@T of type
GroupByAccum<int cid, SumAccum<float> revenue>

define F(edge b of type 'Bought' from c to p) {
    if p.category = 'Toys' then
       @@T += (c.cid -> b.quantity * p.price)
    end
}

for each edge e of type 'Bought' do in parallel
    F(e)
end
```

In the above pseudocode, we borrow the GSQL syntax for signalling that an identifier denotes a global accumulator: prepend two @ characters to its name. Thus, @@T is a global accumulator called T. Recall that globality means that there is only one instance of @@T for the entire query.

We also borrow from GSQL the syntax for inserting a key-value pair (k, v) into a group-by accumulator @@*acc*: `@@acc += (k -> v)`. Notice the function F, which is applied to individual edges. F inserts the revenue from the individual sale modeled by the b edge into the nested *revenue* accumulator associated to customer c's *cid* attribute ($c.cid$): `@@T += (c.cid -> b.quantity * p.price)`.

The for loop maps F over all relevant edges in parallel. The order in which the various invocations of F write into @@T is not determined, depending on how they interleave at runtime. Nevertheless, the semantics of the plan is well-defined, as the result is race-free and interleaving-invariant. Race freedom is ensured via an exclusive locking mechanism that all invocations of F use to write into @@T. Interleaving invariance follows from the fact that @@T aggregates product prices using addition, which is commutative and associative.

When the plan's execution completes, the result is centralized in the global accumulator @@T. GSQL allows subsequent query blocks to refer to the table T, in which case their plan accesses the contents of accumulator @@T instead.

6 Changing Grouping Criteria to Increase Parallelism

While the EMVR plan in Section 5 shows that tables can be supported using accumulators, we observe that this plan features only limited potential for parallel evaluation. Despite the function F being mapped in parallel over all relevant edges, each write operation to global accumulator @@T involves the acquisition of an exclusive lock to avoid race conditions. This effectively serializes the invocations of F, turning the operation of writing to the global accumulator into a bottleneck.

Parallelism can be dramatically unlocked by observing that, since attribute *cid* is a key for *Customer* vertices, each group corresponds to precisely one such vertex. This enables the removal of the global accumulator bottleneck @@T by replacing it with many vertex-attached accumulators, one located at each *Customer* vertex. By writing only into its own accumulator, each vertex can work on its own group independently of the other vertices, yielding an eminently parallelizable plan. We detail this alternate plan below.

```
attach to each 'Customer' vertex its own
accumulator of type SumAccum<float>, called @revenue

define F_opt(edge b of type 'Bought' from c to p) {
    if p.category = 'Toys' then
       c.@revenue += b.quantity * p.price
    end
}

for each edge e of type 'Bought' do in parallel
    F_opt(e)
end
```

Following GSQL syntax, the single leading @ indicates that the *@revenue* accumulators are vertex-attached. The type of these accumulators, `SumAccum<float>`, states that they each take floating-point inputs and add them to their stored value.

Also following GSQL syntax, *c.@revenue* denotes the *@revenue* accumulator attached to vertex c.

Notice that the edge function F_{opt} writes into *c.@revenue*, the accumulator of *Customer* vertex c. Only the writes due to edges adjacent to c compete for access to this accumulator and require serialization. Thus, each *@revenue* accumulator serializes only as many writes as the degree of c, which is likely a much smaller quantity than the number of writes serialized by the global accumulator $@@T$, namely the total count of *Bought* edges in the graph.

Upon completion of the plan's execution, there is no centralized data structure holding table T. This table is virtual; its contents are distributed across *Customer* vertices. The components of each virtual tuple $t \in T$ resides a the corresponding vertex c, with *t.cid* stored in *c.cid* and *t.revenue* stored in *c.@revenue*.

7 Reduction to Relational Minimization under Constraints

We present an optimization technique that reasons about the structure of the graph to automatically rewrite global-accumulator-based plans (like the one in Section 5) into vertex-accumulator-based plans (like the one in Section 6) whenever possible.

The crux of the reasoning consists in deciding whether the original group-by list can be replaced with a single vertex variable while preserving the partition of the binding table into groups. If this is possible, then the output table can be virtualized and distributed as illustrated in Section 6. In not possible, groups do not correspond to individual vertices and therefore need to be hosted somewhere else; in the EMVR model, the only alternative is a global accumulator (as illustrated in Section 5). The global-accumulator-based plans are always am available choice, and in the general case may be unavoidable. However, they restrict parallelism, so we seek alternatives whenever possible.

It turns out that we need not devise a customized algorithm for this optimization task. Instead, we can reduce it to the well-studied problem of minimization of relational queries under integrity constraints, for which there exist sound and complete algorithms [6, 10]. The technically interesting exercise consists in defining the reduction, which we present below.

7.1 Encoding vertex types as relations

We encode each vertex type VT as a homonymous relational table, whose columns correspond to VT's attributes. We also prepend a column intended to hold the object id of the vertex.

▶ **Example 3.** In our example, assuming that the *Customer* vertex type has attributes *cid*, *name* and *address*, it is encoded as a table

$$Customer(id, cid, name, address).$$

Similarly, assuming that the *Product* vertex type has attributes *pno*, *pname*, *price* and *category*, we encode it as table

$$Product(id, pno, pname, price, category).$$ ⌟

7.2 Encoding edge types as relations

We encode each edge type ET as a homonymous relational table whose columns correspond to ET's attributes. We also prepend two columns, intended to hold the ids of the edge's source and target vertices.

▶ **Example 4.** In our example, assuming that the *Bought* edge type has attributes *timestamp* and *quantity*, it is encoded as a table

$$Bought(c, p, timestamp, quantity).$$ ⌟

7.3 Encoding the group-by list as a relational query

The list of group-by attributes is the one we seek to rewrite. Since the problem we reduce to is query minimization under constraints, we encode this list as a relational query whose body contains the relational encoding of the type information for the variables of the original query's graph pattern.

▶ **Example 5.** We show the encoding of the group-by list of our running example query as a relational conjunctive query:

$$\begin{array}{rcl} G() & \leftarrow & GroupBy(cid), Customer(c, cid, name, address), \\ & & Bought(c, p, timestamp, quantity), Product(p, pno, pname, price, category). \end{array}$$

Notice the use of the predicate *GroupBy* to label the elements of the group-by list. We seek to rewrite this to

$$\begin{array}{rcl} G() & \leftarrow & GroupBy(c), Customer(c, cid, name, address), \\ & & Bought(c, p, timestamp, quantity), Product(p, pno, pname, price, category) \end{array}$$

which groups by the vertex id instead of the *cid* attribute. ⌟

Note that the encoding queries in Example 5 are boolean (they have no distinguished attributes). A natural encoding alternative that we initially considered would have chosen to place the group-by list elements as distinguished variables in the query head ($G(cid)$ and $G(c)$ in the example). However, this would have potentially required the rewriting to change the type of the query output: for instance, assume alphanumerical strings for the *cid* attribute, and unsigned integers for the vertex id. More gravely, even if we ignore typing information, there are cases when the original group-by list has a different length than the rewritten one, requiring a change in the arity of the query head. Standard query minimization would no longer apply in this case, as it always preserves the arity of the query.

▶ **Example 6.** Assuming that persons are identified by their name, date of birth, and place of birth, and that the original query seeks to group by persons, the alternate encoding scheme we considered would yield a query like

$$G(name, dob, place) \leftarrow Person(p, name, dob, place).$$

We seek to rewrite the group-by list to use the vertex id instead. In relationally encoded form, this would amount to obtaining

$$G(p) \leftarrow Person(p, name, dob, place)$$

which would technically fall outside the purview of standard query minimization. ⌟

The reason why this more natural encoding disables standard query minimization is fundamental: minimization preserves equivalence to the original query, and the concept of equivalence between queries with different output types/arity seems meaningless. It is conceivable (and we were tempted!) to define a more relaxed version of query equivalence (and therefore minimization) that changes the arity of the query head and it would be possible to adapt state-of-the-art algorithms [6] accordingly. The benefit of the alternate encoding introduced here is that it reuses the standard theory and algorithms with no change.

▶ **Example 7.** Revisiting Example 6, our actually adopted encoding scheme yields

$$G() \leftarrow GroupBy(name), GroupBy(dob), GroupBy(place), Person(p, name, dob, place)$$

for the original group-by list, and

$$G() \leftarrow GroupBy(p), Person(p, name, dob, place)$$

for its rewritten form.

Notice how this encoding into boolean queries avoids the issue of arity and/or type discrepancy between original and rewritten query output. ⌟

7.4 Encoding the vertex id using relational dependencies

The property graph data model regards vertices through an object-oriented lens: each vertex is an object with an id that uniquely identifies it. This fact is crucially exploited in rewriting the group-by list and therefore must be captured in the relational encoding.

The consequence for the encoding is that the id column acts as a key for the table modeling the vertex type. The standard way to express key constraints in a relational setting is to resort to *functional dependencies* [2].

▶ **Example 8.** We can express the fact that the id of Customer vertices uniquely determines each vertex, in particular the values of the vertex attributes, using the functional dependency

$$\begin{aligned}&Customer(c, cid_1, name_1, address_1) \wedge Customer(c, cid_2, name_2, address_2)\\ &\longrightarrow cid_1 = cid_2 \wedge name_1 = name_2 \wedge address_1 = address_2.\end{aligned}$$ ⌟

Functional dependencies like the one shown in Example 8 are useful in optimizing the body of the query reflecting from the graph patterns in the `FROM` clause and the conditions in the `WHERE` clause. This kind of optimization is beyond the scope of this paper, as it can be tackled by standard rewriting techniques such as the ones in [6, 10]. Here, we focus on optimizing the group-by list. Unfortunately, this optimization requires a kind of rewriting that standard algorithms for minimization under constraints cannot achieve using functional dependencies alone.

Intuitively, the reason is that the rewriting needs to drop certain atoms from the query body and introduce new ones instead: in Example 7, atoms $GroupBy(name)$, $GroupBy(dob)$, $GroupBy(place)$ need to be dropped, and atom $GroupBy(p)$ added; in Example 5, atom $GroupBy(cid)$ needs to be dropped in favor of adding atom $GroupBy(c)$. These operations require as justification a class of relational dependencies known as *tuple-generating dependencies (TGDs)* [2]. We therefore add to the relational encoding appropriate TGDs.

▶ **Example 9.** The fact that the id of Customer vertices uniquely determines each vertex, in particular the values of the vertex attributes, implies that, once the query groups by the vertex id, it can just as well group by any of the vertex attributes without changing the contents of the groups. We capture this with the following TGD:

$$\begin{aligned} Customer(c, cid, name, address) \wedge GroupBy(c) &\rightarrow GroupBy(cid) \\ &\wedge\ GroupBy(name) \\ &\wedge\ GroupBy(address) \end{aligned}$$ ⌟

7.5 Encoding vertex key constraints as relational dependencies

Another crucial ingredient on which the reasoning about the group-by list is based is the knowledge about vertex keys, i.e. attribute sets whose values uniquely determine the vertex. Vertex keys are analogous to their relational counterpart, the relational key integrity constraint. And yet, for our purposes the encoding of vertex key constraints requires TGDs, in contrast to the standard representation of relational keys using functional dependencies. The motivation is analogous to the one discussed for encoding knowledge about vertex ids.

▶ **Example 10.** In our running example, the *cid* attribute value determines the *Customer* vertex. Consequently, *cid* is a key for the encoding relational table, and this information is typically captured by the functional dependency

$$\begin{aligned} &Customer(c_1, cid, name_1, address_1) \wedge Customer(c_2, cid, name_2, address_2) \\ &\longrightarrow c_1 = c_2 \end{aligned}$$

which states that the *Cid* attribute determines the identity of the *Customer* vertices. As per our previous discussion, this functional dependency does not suffice for rewriting the group-by list, as it needs to drive the introduction and dropping of *GroupBy* atoms. What is needed is a tuple-generating dependency such as

$$Customer(c, cid, name, address) \wedge GroupBy(cid) \longrightarrow GroupBy(c).$$

Intuitively, this TGD states that, if the query already groups by the *cid* attribute, it might just as well group by the vertex id without changing the contents of the resulting groups. ⌟

7.6 Putting it all together: rewriting the group-by list

Our technique for rewriting the group-by list involves the following steps:

1. Encode the group-by list as a relational conjunctive query Q using the types inferred for the variables in the original graph pattern.
2. Encode the vertex id and vertex key information for each vertex type using relational dependencies; call their set Σ.
3. Feed Q and Σ to an algorithm for minimization under constraints (such as the Provenance-Aware Chase&Backchase (PACB) [6, 10]). The algorithm is invoked as a black box, which returns a set of rewritings of Q.
4. Inspect the rewritings, keeping those that contain a single *GroupBy* atom, the argument of which is a variable that binds to vertex identifiers.

Any rewriting found in Step 4 admits a plan in which each group is computed in parallel at its corresponding vertex, and in which the per-group aggregation is implemented by vertex-attached accumulators. In case of multiple qualifying rewritings, the selection can be informed by cost information (cost-based selection is beyond the scope of this paper).

▶ **Example 11.** For the running example, we obtain the first query from Example 5 and the set Σ comprising the dependencies from Examples 8, 9, and 10. The PACB algorithm run on this query with Σ yields two minimal rewritings, corresponding to both queries shown in Example 5. They are both minimal, but only the second query satisfies the criteria of Step 4 above. From this rewriting, we can directly generate a plan that aggregates groups in vertex-attached accumulators at the *Customer* vertices.

Though understanding the inner workings of the PACB algorithm is not necessary since the latter is used as a black box, we sketch them for this example in order for the reader to better appreciate the need to design the encoding in its current form.

The PACB starts from query

$$\begin{array}{lll} G() & \leftarrow & GroupBy(cid), Customer(c, cid, name, address), \\ & & Bought(c, p, timestamp, quantity), Product(p, pno, pname, price, category). \end{array}$$

It chases this query with the vertex-key-encoding dependency

$$Customer(c, cid, n, a) \wedge GroupBy(cid) \longrightarrow GroupBy(c)$$

from Example 10, to obtain

$$\begin{array}{lll} G() & \leftarrow & GroupBy(cid), Customer(c, cid, name, address), \\ & & Bought(c, p, timestamp, quantity), Product(p, pno, pname, price, category), \\ & & \boxed{\text{GroupBy(c)}}. \end{array}$$

Note the addition of the $GroupBy(c)$ atom. This query encodes a group-by list that includes *both* the attribute *cid* and the identifier c of *Customer* vertices.

A further chase step applies, with the vertex-id-encoding dependency

$$\begin{array}{lll} Customer(c, cid, name, address) \wedge GroupBy(c) & \rightarrow & GroupBy(cid) \\ & \wedge & GroupBy(name) \\ & \wedge & GroupBy(address) \end{array}$$

from Example 9, yielding

$$\begin{array}{lll} G() & \leftarrow & GroupBy(cid), Customer(c, cid, name, address), \\ & & Bought(c, p, timestamp, quantity), Product(p, pno, pname, price, category) \\ & & \boxed{\text{GroupBy(c)}}, \\ & & \boxed{\text{GroupBy(name), GroupBy(address)}}. \end{array}$$

Notice the addition of the last two *Groupby* atoms. No further chase steps apply.

Intuitively, the algorithm has at this point inferred that, given that the original query was grouping by the *cid* attribute of *Customer* vertices, it is safe to add the vertex identity and all other *Customer* vertex attributes to the group-by list as this won't change the group contents.

The PACB algorithm now seeks minimal rewritings of the original query among the subqueries of the chase result (see [6] for details on how this search is directed to avoid exhaustively inspecting all subsets of atoms in the chase result). Notice that both queries from Example 5 are found this way. The first query coincides with the original user query (unsurprisingly, as its singleton group-by list is necessarily minimal). More interestingly, it turns out that the second query is an equivalent rewriting (the PACB checks equivalence by chasing, and this time the TGD from Example 9 is instrumental).

Also note that the second query satisfies the requirement of having a single *GroupBy* atom, whose variable c binds to vertex identifiers (namely those of *Customer* vertices). It is therefore the one returned by Step 4 of our optimization technique. ⌟

Let's call a group-by list *Unary Vertex-Grouping (UVG)* if it contains a single variable, which binds to vertex identifiers. We say that a query is UVG if its group-by list is UVG. UVG queries admit plans based on vertex-attached accumulators, thus avoiding the aggregation bottleneck caused by the use of global accumulator to simulate the output table. Our technique is guaranteed sound and complete for finding UVG rewritings.

▶ **Theorem 12.** *The technique described here is guaranteed to find precisely all UVG rewritings of a given query's group-by list under a given set of vertex key constraints.*

This follows from the fact that all TGDs involved in the relational encoding are so-called *full*, for which the chase procedure is guaranteed to terminate [2]. In addition, it follows from the soundness and completeness of the PACB algorithm whenever the chase terminates [6, 10].

7.7 Exploiting Additional Classes of Constraints

Since our approach is based on a reduction to relational minimization under constraints, as long as the constraints in the graph schema admit encoding as dependencies for which the chase terminates, the PACB remains a sound and complete minimization procedure. We can exploit such constraints for free, allowing them to synergistically interact with each other to uncover further UVG rewritings.

One class of frequently encountered constraints pertains to the cardinality of edges when viewed as relationships between the connected nodes. Many-to-one and one-to-one constraints can also be exploited to uncover UVG rewritings of the group=by list.

▶ **Example 13.** Assume that the graph in our running example provides information on the customer's city via a *LivesIn* edge.

Consider the following variation of our running example query, in which we wish to list the customer *cid*, the *name* of the city they live in, and the revenue from the sales to this customer.

```
SELECT   c.cid, cty.name, sum(b.quantity * p.price) AS revenue INTO T
FROM     (cty:City) <-[:LivesIn]- (c:Customer) -[b:Bought]-> (p:Product)
GROUP BY c.cid, cty.name
```

Assume that the graph schema declares, as above, that the attribute *cid* is a key for *Customer* vertices. Additionally, it states that attribute *name* is a key for *City* vertices, and that the *LivesIn* edge is many-to-one (customers live in at most one city). In that case the above query can be equivalently rewritten to

```
SELECT   c.cid, cty.name, sum(b.quantity * p.price) AS revenue INTO T
FROM     (cty:City) <-[LivesIn]- (c:Customer) -[b:Bought]-> (p:Product)
GROUP BY c
```

whose new group-by list makes explicit the fact that each group is determined by a *Customer* vertex. Table T can once again be virtualized and distributed across *Customer* vertices by a plan based on vertex-attached accumulators.

The rewritten group-by list is obtainable by feeding the PACB the dependencies from Example 11, accompanied by the encoding of the key constraint for *City* vertices,

$$City(cty, name, population) \wedge GroupBy(name) \longrightarrow GroupBy(cty)$$

and the encoding as a TGD of the many-to-one constraint on the *LivesIn* edge:

$$LivesIn(cust, cty) \wedge GroupBy(cust) \longrightarrow GroupBy(cty).$$

Intuitively, the latter dependency states that, once the query groups by *Customer* vertex id, it can just as well group by the *City* vertex id without changing the groups, because the *Customer* vertex determines the *City* vertex. ⌟

In general, we capture a many-to-one constraint on an edge of type ET with the TGD

$$ET(src, tgt, \ldots) \wedge GroupBy(src) \longrightarrow GroupBy(tgt),$$

where the dots stand for the edge attributes.

One-to-one constraints on an edge are captured by two TGDs, corresponding to the many-to-one TGDs in each direction.

We can uncover further rewriting opportunities by exploiting any additional class of constraints, as long as they are expressible as relational dependencies, and as long as the chase with the resulting set of dependencies terminates.

8 Conclusion

For graph queries that return tables by performing grouping and aggregation in an engine conforming to the edge-map/vertex-reduce paradigm, the grouping criteria determine the degree of parallelism of the aggregation task. By exploiting the integrity constraints in the graph schema, the user query's group-by list can be equivalently rewritten to expose the maximally available degree of aggregation parallelism. This applies more generally to all models in the class of vertex-centric computation models. There is no need to devise novel algorithms for rewriting the group-by list under integrity constraints - it suffices to devise a novel encoding scheme that reduces the problem to relational query minimization under dependencies. This enables us to leverage the sound and complete PACB minimization algorithm introduced in Peter Buneman's Festschrift [6].

References

1 Serge Abiteboul, Peter Buneman, and Dan Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

2 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

3 Khaled Ammar and M. Tamer Özsu. Experimental analysis of distributed graph systems. *Proc. VLDB Endow.*, 11(10):1151–1164, June 2018. `doi:10.14778/3231751.3231764`.

4 Apache. Apache giraph, 2020. URL: `https://giraph.apache.org/`.

5 Apache. Apache spark graphx, 2020. URL: `https://spark.apache.org/graphx/`.

6 Alin Deutsch and Richard Hull. Provenance-directed chase&backchase. In Val Tannen, Limsoon Wong, Leonid Libkin, Wenfei Fan, Wang-Chiew Tan, and Michael P. Fourman, editors, *In Search of Elegance in the Theory and Practice of Computation - Essays Dedicated to Peter Buneman*, volume 8000 of *Lecture Notes in Computer Science*, pages 227–236. Springer, 2013. doi:10.1007/978-3-642-41660-6_11.

7 Alin Deutsch, Yu Xu, Mingxi Wu, and Victor E. Lee. Tigergraph: A native MPP graph database. *CoRR*, abs/1901.08248, 2019. arXiv:1901.08248.

8 Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 17–30, Hollywood, CA, October 2012. USENIX Association. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez.

9 Minyang Han, Khuzaima Daudjee, Khaled Ammar, M. Tamer Özsu, Xingfang Wang, and Tianqi Jin. An experimental comparison of pregel-like graph processing systems. *Proceedings of the VLDB Endowment*, 7:1047–1058, August 2014. doi:10.14778/2732977.2732980.

10 Ioana Ileana, Bogdan Cautis, Alin Deutsch, and Yannis Katsis. Complete yet practical search for minimal query reformulations under constraints. In Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu, editors, *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1015–1026. ACM, 2014. doi:10.1145/2588555.2593683.

11 ISO. GQL, 2024. URL: https://www.iso.org/standard/76120.html.

12 S. Kazem Lellahi and Val Tannen. A calculus for collections and aggregates. In Eugenio Moggi and Giuseppe Rosolini, editors, *Category Theory and Computer Science, 7th International Conference, CTCS '97, Santa Margherita Ligure, Italy, September 4-6, 1997, Proceedings*, volume 1290 of *Lecture Notes in Computer Science*, pages 261–280. Springer, 1997. doi:10.1007/BFB0026993.

13 Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 340–349, Arlington, Virginia, USA, 2010. AUAI Press.

14 Yi Lu, James Cheng, Da Yan, and Huanhuan Wu. Large-scale distributed graph computing systems: An experimental evaluation. *Proceedings of the VLDB Endowment*, 8, November 2014.

15 Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1807167.1807184.

16 Robert McCune, Tim Weninger, and Gregory Madey. Thinking like a vertex: a survey of vertex-centric frameworks for distributed graph processing. *ACM Computing Surveys*, 48, July 2015. doi:10.1145/2818185.

17 Semih Salihoglu and Jennifer Widom. Gps: A graph processing system. In *Scientific and Statistical Database Management*. Stanford InfoLab, July 2013. URL: http://ilpubs.stanford.edu:8090/1039/.

18 Philip Stutz, Abraham Bernstein, and William Cohen. Signal/collect: Graph algorithms for the (semantic) web. *Lecture Notes in Computer Science (LNCS)*, 6496:764–780, October 2010. doi:10.1007/978-3-642-17746-0_48.

19 Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990. doi:10.1145/79173.79181.

20 Da Yan, Yingyi Bu, Yuanyuan Tian, and Amol Deshpande. Big graph analytics platforms. *Foundations and Trends in Databases*, 7:1–195, January 2017. doi:10.1561/1900000056.

Fishing Fort: A System for Graph Analytics with ML Prediction and Logic Deduction

Wenfei Fan
Shenzhen Institute of Computing Sciences, China
University of Edinburgh, UK
Beihang University, Beijing, China

Shuhao Liu
Shenzhen Institute of Computing Sciences, China

Abstract

This paper reports Fishing Fort, a graph analytic system developed in response to the following questions. What practical value can we get out of graph analytics? How can we effectively deduce the value from a real-life graph? Where can we get clean graphs to make accurate analyses possible? To answer these questions, Fishing Fort advocates to unify logic deduction and ML prediction by proposing Graph Association Rules (GARs), a class of logic rules in which ML models can be embedded as predicates. It employs GARs to deduce graph associations, enrich graphs and clean graphs. It has been deployed in production lines and proven effective in online recommendation, drug discovery, credit risk assessment, battery manufacturing and cybersecurity, among other things.

2012 ACM Subject Classification Information systems → Data mining; Information systems → Data management systems; Information systems → Information integration

Keywords and phrases graph analytics, data cleaning, association analysis

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.6

Acknowledgements The paper is a tribute to Professor Val Tannen. Val was a professor at UPenn when Fan was doing PhD there, and has been providing Fan with unfailing support ever since.

1 Introduction

When working with practitioners in industry, we often hear the following questions.

(1) What practical value can we get out of big data analytics? After a decade of study, people are still scrambling to find killer apps of big data analytics. While the success of machine learning (ML) should be at least partly attributed to big data, people have higher expectations from big data analytics to help them solve problems at hand, e.g., optimize a costly step in a manufacturing process or find the use of an old drug for a new disease. Besides, big data analytics gets credits for making an advance only if it is not entirely ML-based, i.e., it approaches the problem not by simply training and applying an ML model.

(2) How should we analyze big data, ML prediction or logic deduction? Data analytics is often approached by employing either ML models or logic rules. However, neither of the two is superior to the other. On the one hand, it is hard to discover a small number of accurate logic rules to cover different cases of our application and data. On the other hand, ML predictions are probabilistic and hard to explain; practitioners may not want to make critical decisions based on ML predictions alone. Is it possible to unify the two and benefit from both?

(3) Where can we get clean data for analytics? Real-life data is often dirty, as evidenced by duplicates, semantic inconsistencies, stale data and missing values commonly found in practice. Dirty data has been a longstanding challenge. To make practical use of big data analytics, it is a must to clean the data. Indeed, data-driven decisions based on dirty data can be worse than making decisions with no data. For instance, "as a healthcare, retail, or financial services business you cannot afford to make decisions based on yesterday's data" [9].

In an effort to tackle these questions, we have developed Fishing Fort, a system for deducing associations among entities in graphs. Fishing Fort has been deployed in the production lines of several domains, and has proven effective there. The system is named in memory of the siege of Diaoyu (Fishing) Fortress (a.k.a. Diaoyu Castle), a battle during which the fourth khagan of the Mongol Empire was severely wounded to death by the defenders, and a battle that changed the history of China and the world. The system has several unique features.

(1) Unification of ML and logic. Fishing Fort unifies ML prediction and logic deduction by employing a class of Graph Association Rules (GARs) [17]. A GAR is composed of a graph pattern to identify related entities, and a dependency on the entities to disclose their association, correlation and interaction. Taken together, the pattern and dependency reveal regularities among entities in graphs. Moreover, a GAR may embed ML models as predicates. As a consequence, GARs may embed ML predictions in logic deduction, and moreover, explain local predictions of ML models such as graph neural networks (GNNs) [26, 27, 4].

(2) Accurate association deduction. Fishing Fort deduces associations with GARs. It supports algorithms for discovering GARs from real-life graphs [13], conducting (batch and incremental) deduction [17] and making event predictions [18]. It has found successful applications in early drug discovery [11], lithium-ion battery manufacturing, risk control for bank loans, cyber attack detection and online recommendation in e-commerce [10], among other things.

(3) Data enrichment and cleaning. Fishing Fort can also be used to improve the quality of real-life graphs. Given a graph G, it enriches G by extracting data that pertains to the entities of G from external data sources, e.g., knowledge graphs [19]. It detects and fixes errors in the (enriched) graph by employing a special form of GARs [12], in polynomial time (PTIME). The fixes computed are logical consequences of the rules and accumulated ground truth (i.e., validated facts); hence if the rules and ground truth are correct, so are the fixes.

Organization. The remainder of the paper is organized as follows. Section 2 defines GARs. Section 3 presents the architecture of Fishing Fort and showcases how it deduces associations with GARs in real-life applications. Section 4 shows how Fishing Fort enriches and cleans graphs with (a special form) of GARs. Finally, Section 5 discusses future plans.

2 Unifying ML prediction and Logic Deduction

This section reviews basic notations (Section 2.1), and defines Graph Association Rules (GARs) (Section 2.2). It also reports the complexity for reasoning about GARs (Section 2.3).

2.1 Preliminaries

Assume three countably infinite sets of symbols, denoted by Ω, Υ and U, for labels, attributes and constants, respectively. We consider *graphs* $G = (V, E, L, F_A)$, where (a) V is a finite set of vertices, (b) $E \subseteq V \times \Omega \times V$ is a finite set of edges, where each $e = (v, l, v')$ in E denotes an edge from vertex v to v' that is labeled with $l \in \Omega$, (c) L is a function such that for each vertex $v \in V$, $L(v)$ is a label from Ω, and (d) each vertex $v \in V$ carries a tuple $F_A(v) = (A_1 = a_1, \ldots, A_n = a_n)$ of *attributes* of a finite arity, where $A_i \in \Upsilon$ and $a_i \in U$, written as $v.A_i = a_i$, and $A_i \neq A_j$ if $i \neq j$, representing properties. Different vertices may carry different attributes, which are not constrained by a schema.

A *graph pattern* is $Q[\bar{x}] = (V_Q, E_Q, L_Q, \mu)$, where (1) V_Q (resp. E_Q) is a finite set of *pattern nodes* (resp. *edges*), (2) L_Q assigns a label $L_Q(u) \in \Gamma$ to each pattern node $u \in V_Q$, (3) $\bar{x}$ is a list of distinct variables; and μ is a mapping that assigns a distinct variable to each node v in V_Q. For $x \in \bar{x}$, we use $\mu(x)$ and x interchangeably when it is clear in the context.

A *match* of pattern $Q[\bar{x}]$ in graph G is defined as a homomorphism h from Q to G such that (a) for each node $u \in V_Q$, $L_Q(u) = L(h(u))$; and (b) for each pattern edge $e = (u, l, u')$ in Q, $e' = (h(u), l, h(u'))$ is an edge in G. We denote the match as a vector $h(\bar{x})$, consisting of $h(x)$ for all $x \in \bar{x}$ in the same order as $\bar{x}$, denoting entities identified by Q.

2.2 Graph Association Rules

GARs are defined with predicates. A *predicate* of pattern $Q[\bar{x}]$ is one of the following:

$$p ::= l(x, y) \mid x.A \otimes y.B \mid x.A \otimes c \mid \mathcal{M}(x.\bar{A}, y.\bar{B}),$$

where $\otimes$ is one of $=, \neq, <, \leq, >, \geq$; $l \in \Omega$; x and y are variables in $\bar{x}$; c is a constant in U; A and B are attributes; and $x.\bar{A}$ is a list of attributes at "vertex" x; similarly for $y.\bar{B}$.

The predicates are classified as follows. (1) Logic predicates: *link predicate* $l(x, y)$ indicates the existence of an edge labeled l from vertex x to y; *variable predicate* $x.A \otimes y.B$ and *constant predicate* $x.A \otimes c$ check the correlation and interaction of attribute values. (2) ML predicates: $\mathcal{M}(x.\bar{A}, y.\bar{B})$ is a pre-trained ML model that returns true iff $\mathcal{M}$ predicts true at $(x.\bar{A}, y.\bar{B})$. Here $\mathcal{M}$ can be any ML model that returns Boolean (e.g., $\mathcal{M} \geq \sigma$ for a predefined bound σ).

Rules. A GAR (*Graph Association Rule*) is a pair of a graph pattern and a dependency [17]:

$$\varphi = Q[\bar{x}](X \rightarrow p_0),$$

where $Q[\bar{x}]$ is a graph pattern, X is a conjunction of predicates of $Q[\bar{x}]$, and p_0 is a single predicate of $Q[\bar{x}]$. We refer to $Q[\bar{x}]$ and $X \rightarrow p_0$ as the *pattern* and *dependency* of GAR φ, respectively, and to X and p_0 as the *precondition* and *consequence* of φ, respectively.

Intuitively, Q in a GAR identifies entities in a graph, and $X \rightarrow p_0$ is applied to the entities. Predicates $x.A \otimes c$ and $x.A \otimes y.B$ specify *value associations of* attributes. Link predicates enforce edge existence to deduce missing links. Moreover, one can "plug in" pre-trained ML models $\mathcal{M}$ for entity resolution [33], link predictions [44] and similarity checking [8]. For each application domain, Fishing Fort maintains a library of pre-trained ML models.

We can uniformly express various ML models we used as link prediction for $l(x, y)$, where the label l can indicate (1) a predicted link, (2) the match of x and y as the same entity, linked by a dummy edge with "$=$" as l, or (3) semantic similarity between vertices x and y linked by an edge with "$\approx$" as l, indicating that x and y are "semantically" close.

We will showcase GARs for real-life applications in Section 3.

As shown in [17], graph functional dependencies (GFDs) [23], graph entity dependencies (GEDs) [20] and graph pattern association rules (GPARs) [22] are special cases of GARs. GARs extend these graph dependencies by supporting ML and link predicates. Note that link predicates mutate the topological structure of a graph. Besides the primitive predicates above, Fishing Fort also supports, e.g., local 2-dimensional Weisfeiler-Leman (local 2-WL) test [27], to explain GNN-based recommendations and link predictions. It is known that such GNN models are no more expressive than local 2-WL test [27].

Semantics. Consider a GAR $\varphi = Q[\bar{x}](X \rightarrow p_0)$. Denote by $h(\bar{x})$ a match of pattern Q in a graph G, and by p a predicate of $Q[\bar{x}]$. We write $h(\mu(x))$ as $h(x)$, where μ is the mapping in Q from $\bar{x}$ to vertices in Q. A match $h(\bar{x})$ *satisfies* a predicate p, denoted by $h(\bar{x}) \models p$, if one of following conditions is satisfied: (a) when p is $l(x, y)$, there exists an edge with label l from $h(x)$ to $h(y)$; (b) when p is $x.A \otimes y.B$, the vertex $h(x)$ (resp. $h(y)$) carries attribute A (resp. B), and $h(x).A \otimes h(y).B$; similarly for constant predicate $h(x).A \otimes c$; and (c) when p is $\mathcal{M}(x.\bar{A}, y.\bar{B})$, the ML model $\mathcal{M}$ predicts true at $(h(x).\bar{A}, h(y).\bar{B})$.

We write $h(\bar{x}) \models X$ if $h(\bar{x}) \models p$ for *all* p in a set X of predicates. We write $h(\bar{x}) \models X \rightarrow p_0$ if whenever $h(\bar{x}) \models X$, then $h(\bar{x}) \models p_0$. We say that a graph G *satisfies* GAR $\varphi = Q[\bar{x}](X \rightarrow p_0)$, denoted by $G \models \varphi$, if for all matches $h(\bar{x})$ of $Q[\bar{x}]$ in G, $h(\bar{x}) \models X \rightarrow p_0$. We say that G *satisfies* a set Σ of GARs, denoted by $G \models \Sigma$, if for all GARs $\varphi \in \Sigma$, $G \models \varphi$.

For $p = x.A \otimes c$, we use $h(p)$ to denote $h(x).A \otimes c$; similarly for the other predicates. For a set X of predicates, we denote by $h(X)$ the collection of $h(p)$ for all p in X. For a set Γ of validated facts of the forms $u.A \otimes v.B$ and $u.A \otimes c$ for vertices u, v in a graph G, we say that $h(p)$ is *validated* with Γ if it is enclosed in Γ; in particular, for $p = \mathcal{M}(x.\bar{A}, y.\bar{B})$, the values of $h(x).A$ and $h(y).B$ are enclosed in Γ for all corresponding $A \in \bar{A}$ and $B \in \bar{B}$; similarly for $h(X)$. As will be seen in Section 3.1, we check validated facts to deduce reliable associations.

2.3 Complexity

There are three classical problems for dependencies, stated as follows.

The *satisfiability* problem is to decide, given a set Σ of GARs, whether there exists a graph G such that $G \models \Sigma$ and for each GAR $Q[\bar{x}](X \rightarrow p_0) \in \Sigma$, Q has a match in G? Intuitively, this is to ensure that all GARs can be applied to G at the same time without conflicts.

A set Σ of GARs *implies* a GAR φ, denoted by $\Sigma \models \varphi$, if for all graphs G, if $G \models \Sigma$ then $G \models \varphi$, i.e., φ is a logical consequence of Σ. The *implication problem* is to decide, given a set Σ of GARs and a GAR φ, whether $\Sigma \models \varphi$? Intuitively, the implication analysis can help us remove redundant GARs in rule discovery and speed up association analyses with GARs.

The *validation problem* is to decide, given a graph G and a set Σ of GARs, whether $G \models \Sigma$? Intuitively, this is to settle the complexity of association analyses with GARs.

To simplify the implication analysis, we consider ML predicates of the form $\mathcal{M}(x.\bar{A}, y.\bar{B}) \geq \sigma$ for a predefined σ. Moreover, we assume that the ML models in a set of GARs are independent, i.e., they are trained for different purposes and do not overlap/entail each other, e.g., models for predicting the anomaly and capacity of battery cells in battery manufacturing.

It has been shown that the satisfiability, implication and validation are coNP-complete, NP-complete, and coNP-complete for GARs, respectively [17, 20]. Here we assume that given two lists of attributes $x.\bar{A}$ and $y.\bar{B}$, checking $\mathcal{M}(x.\bar{A}, y.\bar{B})$ is in PTIME in the sizes $|x.\bar{A}|$ and $|y.\bar{B}|$, as commonly found in practice for pre-trained $\mathcal{M}$. The complexity bounds are the same as for reasoning about GEDs [20]. Moreover, the implication and satisfiability analyses are no harder than their counterparts for relational conditional functional dependencies [14].

3 Deducing Graph Associations

This section first presents the architecture and workflow of Fishing Fort (Section 3.1), and then showcases its applications in different domains (Section 3.2).

3.1 The Architecture of Fishing Fort

As shown in Figure 1, Fishing Fort supports both association deduction, and graph enrichment and cleaning. For each application domain, it maintains a set Γ of ground truth, and references relevant external knowledge graphs KG. Given a real-life graph G, it first enriches G by extracting relevant data from KG and cleans the enriched G; to simplify the discussion, we refer to the enriched and cleaned graph also as G. After the preprocessing step, Fishing Fort then discovers a set Σ of GARs for the application, and deduces associations from G with the GARs of Σ; in the process it accumulates ground truth for subsequent analyses. Below we focus on association deduction, and defer data enrichment and cleaning to Section 4.

Association deduction is carried out by the following main modules and algorithms.

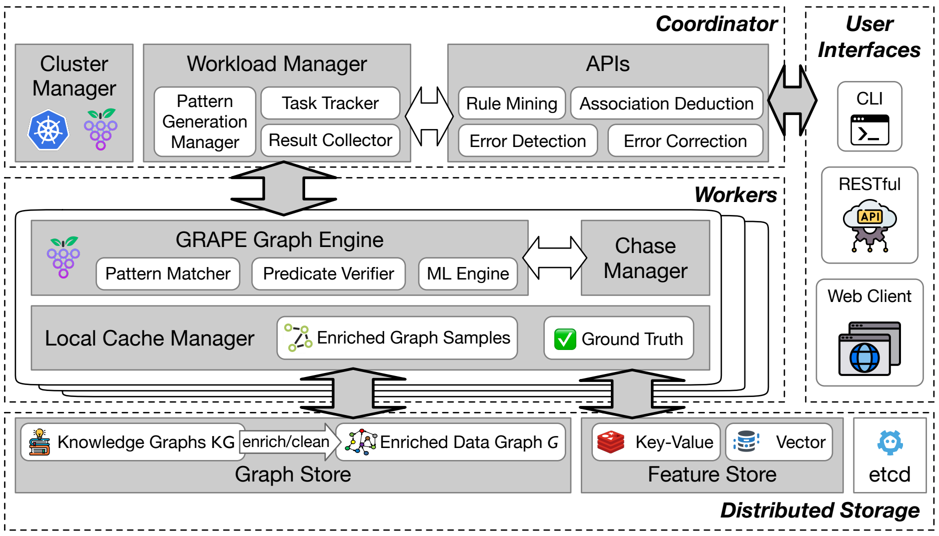


Figure 1 The architecture of Fishing Fort.

Rule discovery. Fishing Fort implements the parallel algorithm of [13] for discovering GARs. To scale with a large graph G, it employs three strategies. (1) Application-driven discovery. Given an application $\mathcal{A}$, Fishing Fort trains an ML model $\mathcal{M}_{\mathcal{A}}$ to identify vertices, edges and properties in G that pertain to $\mathcal{A}$. It reduces G to a smaller graph $G_{\mathcal{A}}$ that consists only of the data pertaining to $\mathcal{A}$. Moreover, it discovers $\mathcal{A}$-*relevant* rules, i.e., GARs with consequence predicates selected by $\mathcal{M}_{\mathcal{A}}$. (2) Sampling. It samples a set H of graphs from $G_{\mathcal{A}}$ such that each graph is at most $\rho\%$ of $G_{\mathcal{A}}$ and consists of representative entities in $G_{\mathcal{A}}$ and their surrounding subgraphs, for a predefined bound $\rho\%$. Denote by Σ_G and Σ_H the set of $\mathcal{A}$-relevant rules discovered from G and H, respectively. Fishing Fort guarantees that given bounds σ and $\gamma\%$, it samples H such that (a) at least $\gamma\%$ of rules in Σ_G are covered by Σ_H, and (b) each of these rules can be applied at least σ times on the entire G, i.e., the rules in Σ_H have recall above $\gamma\%$ and support above σ over the original graph G. (3) Parallel scalability. Fishing Fort employs data-partitioned parallelism. It partitions G and distributes the fragments across different machines. It works on the fragments in parallel and guarantees to reduce runtime when more machines are used, i.e., the parallel scalability [29]. Hence in principle, it can scale with large graphs G by using more machines when needed.

Association deduction. Fishing Fort deduces link and value associations from graph G, by chasing G with the set $\Sigma = \Sigma_H$ of GARs discovered. More specifically, it applies a GAR $Q[\bar{x}](X \to p_0)$ to G only if there exists a match h of Q in G such that $h \models X$ and the $h(X)$ is validated with the ground truth in Γ (see Section 2.2); if so, it deduces association $h(p_0)$ and adds $h(p_0)$ to Γ as validated facts for subsequent deduction. To carry this out, it implements a combination of the parallel algorithms of [17, 21], and supports two modes: (1) batch mode, to deduce all associations from G at once, and (2) incremental mode, to deduce changes online to associations in response to updates to G. In both modes, users may also opt to pick entities of their interest and deduce associations pertaining to these entities only. Moreover, it explains ML predictions if requested by users (see Section 3.2.1).

Fishing Fort provides the following performance guarantees. (a) The chase is Church-Rosser [1], i.e., it guarantees to terminate and converge at the same result no matter what GARs in Σ are used and in what order the rules are applied. (b) The deduced associations are logical consequences of the GARs of Σ and ground truth of Γ; hence if Σ and Γ are correct, so are the deduced associations. (c) Its deduction algorithm is also parallelly scalable.

Implementation. Fishing Fort is implemented on top of GRAPE [24], a graph engine that parallelizes single-machine graph algorithms based on a fixpoint model in terms of partial evaluation and incremental computation. GRAPE was acquired by Alibaba Group and renamed as GraphScope [16]; it supports 90+% of daily graph computations of Alibaba.

3.2 Fishing Fort in Action

We next present example applications of Fishing Fort, along with GARs discovered from real-life data. The graph patterns of the GARs are depicted in Figure 2.

3.2.1 Online Recommendation

ML models have been widely used in e-commerce to recommend items to users. The models are often classified as collaborative filtering (CF) and content-based (CB). CF identifies user preference and makes recommendation by learning from user-item historical interactions, e.g., users' previous ratings and browsing history, and CB primarily compares the contents of users and items such as user profiles and item features. However, a single strategy, either CF or CB, often does not suffice in practice. For example, instead of exploring new interesting items, CF tends to find similar ones w.r.t. the user's past interaction due to its collaborative nature. It does not work well when the interaction data is sparse and when a recommender system starts cold. To rectify these limitations, hybrid models have been explored to unify interaction-level similarity and content-level similarity. However, the hybrid approach often requires training a new ML model starting from scratch. Moreover, the ML models usually compute a prediction score, i.e., recommendation strength, between each user x and item y, and recommend y to x only if the score is above some predetermined thresholds. As a result, these ML models often make false positive (FP) or false negative (FN) predictions, especially in the "fuzzy area" where the predicted scores are near the thresholds, i.e., an area where the prediction strengths are neither sufficiently large nor sufficiently small.

Another issue concerns the explanation of ML predictions $\mathcal{M}(x, y)$, to tell why an item y is recommended to user x. The need for this is twofold, (a) to provide the users with insights and establish their trust in the predictions, and (b) help developers debug ML models by revealing errors or bias in training data that result in adverse and unexpected behaviors.

Fishing Fort has been deployed at e-commerce companies to reduce FPs and FNs of ML recommendation models, and provide local explanations of GNN-based recommendations. It improves the average click-through rate of the companies by up to 50×.

(a) Reducing FPs and FNs of ML predictions. Consider an ML model $\mathcal{M}_{\mathsf{rate}}(x, y)$ for movie recommendations that, given a movie y and a user x, outputs a numeric score, i.e., recommendation strength, between $(0, 1)$. The higher the score, the stronger the confidence. Take 0.5 as the strength threshold, i.e., $\mathcal{M}_{\mathsf{rate}}$ recommends y to x if $\mathcal{M}_{\mathsf{rate}}(x, y) \geq 0.5$.

As an example, GAR $\varphi_1 = Q_1[\bar{x}](X_1 \wedge 0.5 \leq \mathcal{M}_{\mathsf{rate}}(x_0, x_1) \leq 0.6 \rightarrow \mathsf{dislike}(x_0, x_1))$ corrects some FPs of ML model $\mathcal{M}_{\mathsf{rate}}$ in its "fuzzy area" of recommendations. Together with Q_1, precondition X_1 specifies the following: (1) user x_0 disliked a high-rating movie x_2, which stars the same actor x_4 as movie x_1 does; and (2) x_0 also disliked another high-rating movie x_3, which is directed by the director x_5 of movie x_1. Intuitively, model $\mathcal{M}_{\mathsf{rate}}$ (especially if it is CF-based) is inclined to make the recommendation of movie x_1 to user x_0, since many other users could have positive interactions with movies x_1, x_2 and x_3 (hence the high ratings). However, GAR φ_1 overrides the prediction with logic conditions X_1, because unlike an average user, user x_0 seems to approve neither the leading actor nor the director of movie x_1. Among the matching cases of Q_1 in real-world datasets, this rule reduces FPs by 92.3%.

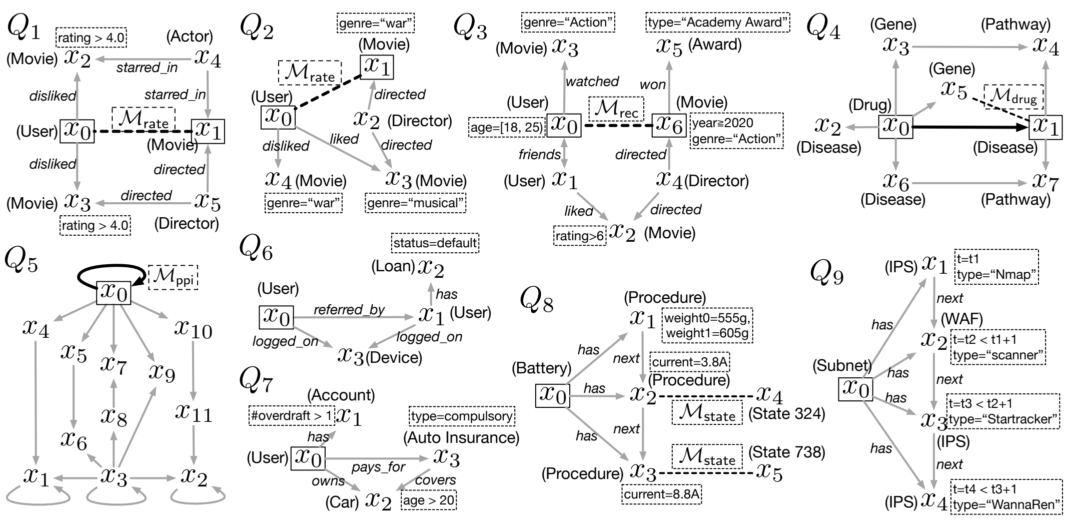


Figure 2 Graph patterns in real-life GARs.

In contrast, GAR $\varphi_2 = Q_2[\bar{x}](X_2 \wedge 0.4 \leq \mathcal{M}_{\mathsf{rate}}(x_0, x_1) \leq 0.5 \rightarrow \mathsf{like}(x_0, x_1))$ suggests recommending a war movie x_1 to user x_0 regardless of the negative ruling of $\mathcal{M}_{\mathsf{rate}}(x_0, x_1)$, if the following conditions (Q_2 and X_2) are met: (1) x_0 disliked a war movie x_4, yet (2) x_0 liked a musical x_3 directed by x_2, the director of movie x_1. Model $\mathcal{M}_{\mathsf{rate}}$ may infer that x_0 is not a fan of war movies based on the interaction with x_4. Nonetheless, rule φ_2 rejects the prediction, based on the evidence that x_0 is likely a fan of director x_2. Over verification datasets, GAR φ_2 reduces FNs of $\mathcal{M}_{\mathsf{rate}}$ by 87% for the matching cases of "fuzzy" predictions.

(b) Explaining ML predictions. Consider $\varphi_3 = Q_3[\bar{x}](X_3 \rightarrow \mathcal{M}_{\mathsf{rec}}(x_0, x_6))$, where $\mathcal{M}_{\mathsf{rec}}$ is a GNN-based model, $\mathcal{M}_{\mathsf{rec}}(x_0, x_6)$ indicates that the model recommends movie x_6 to user x_0, and X_3 is $x_0.\mathsf{age} \geq 18 \wedge x_0.\mathsf{age} < 25 \wedge$ $x_2.\mathsf{rating} \geq 6 \wedge x_3.\mathsf{genre} =$ "Action" $\wedge x_6.\mathsf{genre} =$ "Action" $\wedge x_6.\mathsf{year} \geq 2020$ $\wedge x_5.\mathsf{type} =$ "Academy Award". As opposed to the previous rules, the consequence predicate of this GAR is an ML predicate. Fishing Fort discovers pattern Q_3 and precondition X_3 to explain $\mathcal{M}_{\mathsf{rec}}$ predictions from the movie-watching history and friendship of x_0, and highlights important features of x_0 and x_6.

As a concrete example, suppose that the GNN model $\mathcal{M}_{\mathsf{rec}}$ recommends a movie y_0 "Everything Everywhere All at Once" to a user x_0 "Mike". Then φ_3 may provide high-level rationale behind the recommendation as follows: the movie is recommended to Mike because Mike is a young adult and has watched action movies before, the movie won the 95th academy award for best picture, and it is directed by "Daniel Scheinert", the director of a favorite movie of a friend Peter of Mike. This provides a *local explanation* of the prediction.

Unlike previous methods for explaining ML predictions, e.g., SubgraphX [47] and GNNExplainer [46], Fishing Fort not only provides a subgraph as an explanation, but also tells us what features are decisive for an ML model to make predictions and under what conditions the predictions can be made. Moreover, Fishing Fort supports a predicate of local 2-WL test, and therefore, is able to explain common GNN-based recommendations, which is no more expressive than local 2-WL test [27]. In this way, Fishing Fort deduces local explanations for individual instances with GARs. Moreover, the rules for a GNN-based recommendation model $\mathcal{M}$ disclose the general behaviors of $\mathcal{M}$ and can serve as global explanations of $\mathcal{M}$.

3.2.2 Early Drug Discovery

Drug discovery is the process of new drug identification, which starts from target selection and validation, through preclinical screening, to clinical trials [25]. It is time-consuming and quite expensive. The development of a new drug takes 10–15 years, costs around 2 billion US dollars, and typically has a high risk of failure (>90%) [50].

To accelerate drug discovery, reduce the cost, and increase the success rate, ML models have been explored to identify drug-disease associations (DDAs), drug-drug interactions (DDIs), and protein-protein interactions (PPIs). However, it is costly to train deep-learning models for accurate predictions, and the predictions often have false positives and false negatives. Moreover, ML predictions "still lack reliable explanations that are crucial to drug repurposing and ADR (adverse drug reaction)" [50]. Worse still, noise is often introduced by "unavoidable inaccuracy" of the ML models, e.g., nonexistent relations and inaccurately named entities. Add to the complication that the models are often trained on possibly dirty data, especially when the data comes from multiple sources.

Fishing Fort has been adopted in the early stage of drug discovery. It is used to assist (1) target identification, to identify the right biological molecules or cellular pathways that can be modulated by drugs to achieve therapeutic benefits, where PPIs are often crucial, (b) drug repurposing, to reuse existing drugs for a new disease, and (c) ADR, to disclose undesirable impacts that do not meet the anticipated therapeutic effects and may cause injuries to patients. The need for these is evident. For example, target identification is perhaps the most crucial initial step in drug discovery, and influences the chances of success at every step of drug development. Traditional target identification usually starts in an academic setting, and takes from years to decades. Fishing Fort helps select candidate targets and speed up the process. It identifies DDAs, DDIs and PPIs simply as missing links. Moreover, it has been used to integrate and clean biomedical libraries and data banks (see Section 4).

As an example, Fishing Fort was used to discover GARs for repositioning of existing drugs that may have therapeutic effect on a particular type of Parkinson disease. A GAR is $\varphi_4 = Q_4[\bar{x}](X_4 \rightarrow l(x_0, x_1))$. Together with Q_4, precondition X_1 specifies the following: (1) drug x_0 has a known effect on an inborn genetic blood disease x_2; (2) disease x_1 is the Parkinson; (3) drug x_0 interacts with a gene x_3, which shares an effect pathway x_4 with x_1; (4) drug x_0 can interact with a gene x_5, which has an $\mathcal{M}_{\mathsf{drug}}$-predicted relationship with x_1 (the dashed arrow in Q_4), where $\mathcal{M}_{\mathsf{drug}}$ is a pre-trained ML model that predicts the associations between genes and diseases [40, 34, 44]; and (5) drug x_0 has a known effect on a type of skin cancer x_6, which shares an effect pathway with x_1. The predicted link $l(x_0, x_1)$ (the bold line in Q_4) indicates that drug x_0 may have impact on Parkinson x_1 in some way.

Such GARs suggest five drugs that may have a hidden association with the particular type of Parkinson's disease, including Colforsin (Forskolin) [53], Sulindac [36, 6], Tamoxifen [32], and Tretinoin [43]. Our partners in pharmacy have verified four predictions with published evidence. The remaining one is undergoing their active lab investigation.

As another example, Fishing Fort discovered GARs for PPIs. Such a GAR is $\varphi_5 = Q_5[\bar{x}](\mathcal{M}_{\mathsf{ppi}}(x_0, x_0) \geq \delta \wedge X_5 \rightarrow l(x_0, x_0))$, in which $\mathcal{M}_{\mathsf{ppi}}$ is an RGCN model [37] that predicts PPIs, δ is a prediction threshold, where each vertex in Q_5 denotes a protein and each edge denotes a known PPI, and X_5 specifies additional logic conditions e.g., the domains and subcellular locations, on proteins in Q_5. This GAR identifies the self-interaction $l(x_0, x_0)$ on x_0. It has found a self-interaction on protein SYT2 (the bold line in Q_5) in May 2022, which coincides the finding published in Nature in the same month [31].

When the consequence p_0 of a GAR is an ML model for DDA, DDI or PPI, Fishing Fort can discover pattern Q and conditions X to explain the ML prediction, along the same lines as what we have shown in Section 3.2.1 for local explanations of online recommendation.

3.2.3 Credit Risk Assessment in Banking

Credit risk assessment is crucial for a bank in making decisions on applications for loans. It is the process of evaluating the likelihood that a borrower will default on a loan obligation. It helps minimize the risk of loss from defaults while also making credit available to qualified borrowers, thereby ensuring the bank's profitability and long-term sustainability.

In practice, a bank can use credit scoring models to rate borrowers based on their personal financial information, e.g., credit history, cash flows, and debt levels. In addition, it is equally important to look for other indicators, e.g., the borrower's social interactions, which may expose risks in financial health and the potential inability to repay a loan. However, this requires extensive labor from human experts. One of our FinTech partners employs a team of 1,000+ data analysts, dedicated to disclosing such hidden indicators.

Fishing Fort provides an automated and comprehensive approach. Better yet, it produces indicators that are highly explainable. On graphs G that model user/account interactions, Fishing Fort mined various GARs to assess user risks. A discovered GAR is $\varphi_6 = Q_6[\bar{x}](X_6 \wedge \mathcal{M}_{\mathsf{risk}}(x_0) = \mathsf{low} \rightarrow x_0.\mathsf{risk} = \mathsf{true})$, where $\mathcal{M}_{\mathsf{risk}}(x)$ is a credit scoring model that evaluates user x's default risk as either low, moderate or high, and X_6 is the conjunction of the following conditions: (1) a new user x_0 is referred by user x_1, who failed to repay a loan x_2; and (2) x_0 and x_1 shared a device x_3 for online banking. The rule suggests that user x_0 is likely a high-risk borrower if both pattern Q_6 and precondition X_6 are satisfied, despite the low-risk prediction made by model $\mathcal{M}_{\mathsf{risk}}$. Intuitively, by taking into account clear evidence of straw borrowing, GAR φ_6 corrects some false negatives of $\mathcal{M}_{\mathsf{risk}}$ predictions.

Another GAR $\varphi_7 = Q_7[\bar{x}](X_7 \rightarrow x_0.\mathsf{risk} = \mathsf{true})$ states that user x_0 is likely to default on a loan if x_0 meets the following conditions specified by pattern Q_7 and precondition X_7: (1) x_0 over-drafted bank account x_1 multiple times within the past year; (2) x_0 owns a car x_2 that is 20+ years old; and (3) x_0 is paying only minimum, compulsory liability insurance x_3 for the car. It would be difficult for a human expert to discover such a strong indicator of default risk, since any single factor alone constitutes a mere weak association.

Our partner in personal banking has verified the effectiveness of the GARs mined by Fishing Fort over real-world borrower data. Both φ_6 and φ_7 have a lift over 3, whereas the average lift of an expert-written rule is 2. These rules have been deployed at several banks because of their competitive performance in identifying high-risk borrowers.

3.2.4 Lithium-ion Battery Manufacturing

An electric vehicle (EV) battery pack consists of thousands of lithium-ion cells. These cells must have a balanced capacity, since imbalanced cells may reduce EV range and lead to safety issues such as thermal runaway. An essential phase in the battery manufacturing process consists of two steps: (a) battery formation, which activates a cell by performing the initial charge/discharge operation (to form special electrochemical solid electrolyte interphase at the electrode), followed by battery standing (to stabilize the voltage and cool off); and (b) battery grading, to fully discharge/recharge the cell and test its capacity. This phase is costly, taking from 14 to 20+ hours at different production lines in the industry, and demands specialized devices. It is energy-consuming; per estimation in [48], reducing the battery charge energy by half can save the production line \$65,000 per GWh capacity (assuming a price of \$0.13/kWh). It accounts for more than one third of the total cost of the Lithium-ion battery cell manufacturing process, and is considered one of the bottlenecks that hinders battery manufacturers from increasing output and reducing the total cost of battery production. Therefore, the Battery Formation and Grading System Market has been singled out as a specialized industry segment [30]. It is crucial for the entire Lithium-ion battery industry, including batteries for EVs, electronics and energy storage, among other things.

Fishing Fort provides a cost-effective solution. Traditionally the formation stage first activates the cell by charging it to a ratio of capacity and then cooling it off for a period of time. Grading then fully discharges it, recharges it to its maximum capacity, and discharges it again at a specific rate. It categorizes the capacity of the cell and its performance based on its individual characteristics. As opposed to full discharge/recharge in the traditional process, Fishing Fort only partially charges/discharges a cell. For example, it charges the cell to 50% in the formation stage to activate the cell, and collects data in the process; it then either grades the cell based on the data without discharging/recharging it, or fully discharges it and then partially charges it to a certain voltage for grading, to improve the accuracy. By analyzing the data together with data collected from earlier stages prior to formation, Fishing Fort can accurately grade the capacity of the cell and moreover, determine whether the cell is anomalous or not.

To this end, Fishing Fort first discretizes the time-series measurements; it splits the entire process into consecutive procedures based on their charging/discharging current. It models the data as a graph G with three types of vertices: (1) Procedure carries an array of attributes including the procedure ID, its initial/final weights, the initial/final battery state statistics, and the charging/discharging current; (2) State denotes a state in $\mathcal{S}$; and (3) Battery carries metadata of a battery cell, e.g., the cell ID, the testing slot ID, and its capacity interval. An edge denotes a transition between procedures, an association between a battery cell and a procedure (with range constraints), or an edge between a procedure and a state.

On such a graph G, Fishing Fort discovered GARs for capacity grading. A (simplified) GAR is $\varphi_8 = Q_8[\bar{x}](X_8 \rightarrow x_0.\mathsf{capacity} = 8)$, where its consequence grades a matching battery cell as Capacity Interval 8. Together with Q_8, X_8 specifies the following conditions: (1) the weight before and after the Electrolyte Filling procedure (x_1) is 555 ± 25g and 605 ± 25g, respectively; (2) its Formation-A procedure (x_2) uses a constant charging current at 3.8A, with initial voltage between 0–100mV and a final state 324 (x_4) categorized by $\mathcal{M}_{\mathsf{state}}$ (dashed arrows in Q_8); and (3) its Formation-B procedure (x_3) uses a constant charging current at 8.8A, with initial voltage between 3.3–3.4V and final state 738 (x_5). Intuitively, this rule grades the capacity of a battery cell based on the data collected from partial charge/discharge and from prior stages. It does not require full charge/discharge for the grading step.

In real production, about 3% of the cells have abnormal capacity and thus, should not be packed and assembled with other cells. Fishing Fort also detects anomalous cells online with GARs. It trains an ML model $\mathcal{M}_a$, a binary classification model based on LightGBM for detecting anomaly [28]. It embeds $\mathcal{M}_a$ in the GARs as a predicate, and filters false positives and false negatives of the predictions of $\mathcal{M}_a$ by adding logic predicates to the precondition X. Such a (simplified) GAR is $\varphi'_8 = Q'_8[\bar{x}](x_1.\mathsf{pt}_{10} \leq 1.9 \wedge x_1.\mathsf{pt}_{14} \geq 86 \wedge \mathcal{M}_{\mathsf{a}}(x_1) = \mathsf{false} \rightarrow x_0.\mathsf{status} = \mathsf{anomalous})$, where pattern Q'_8 is similar to Q_8 but does not contain vertices of State, and $x_1.\mathsf{pt}_{10}$ and $x_1.\mathsf{pt}_{14}$ are the values of the voltage and the temperature of cell x_0, respectively. GAR φ'_8 overrides the incorrect negative rulings of model $\mathcal{M}_{\mathsf{a}}$ by directly considering erratic measurements, thus effectively reducing the FNs of $\mathcal{M}_{\mathsf{a}}$.

With such GARs, Fishing Fort reduces charging to 35–50% of the full battery capacity, 75–100% of discharge (in some cases it avoids the grading step completely), and the time for the capacity grading process from 14 hours to 4 hours. With the data of partial charge/discharge, it keeps the error rate under 0.4%, a record in the industry. These translate to an 80% reduction in energy consumption for charging and cooling, cutting equipment costs by half.

3.2.5 Cyber Attack Detection

Cyber attack detection is increasingly vital to enterprise cybersecurity. It aims to identify, analyze, and mitigate threats that could compromise the integrity, confidentiality, and availability of a network. To this end, a common practice is to employ various specialized hardware devices for threat detection and prevention, while ensuring compliance with security standards and regulations. For example, an enterprise network may deploy multiple instances of (a) Web Application Firewalls (WAF), which monitor, filter and block malicious traffic to and from a Web application, and (b) Intrusion Prevention Systems (IPS), which examines network traffic flows to detect and prevent vulnerability exploits.

However, such traditional cybersecurity devices have inherent limitations. First, they operate by identifying and mitigating known threats through predefined policies for access control, signatures for byte-level packet inspection and pattern matching, and detection mechanisms for anomalies; as a result, they can only respond *reactively* to security threats and lack the ability to take *proactive* measures. Second, they often make false positive (legitimate traffic identified as malicious) and false negative (malicious traffic not detected) predictions. This can lead to legitimate users being blocked or attackers slipping through defenses. Worse yet, each single device can only detect a subset of threats depending on its type, configuration and deployed location. Attackers may devise sophisticated, distributed and coordinated attacks to evade detection, leading to increasing false negative rates.

Overcoming these limitations requires advanced *threat intelligence* techniques, which gather and analyze information about emerging threats from various sources around the network. Fishing Fort approaches this by means of temporal graph analytics [18]. By integrating event-level threat intelligence feeds from all devices, Fishing Fort can synthesize security events in real-time, effectively discovering vulnerabilities and new attack vectors. It complements hardware-based solutions with a global view over the entire enterprise network.

As an example, consider GAR $\varphi_9 = Q_9[\bar{x}](X_9 \rightarrow x_0.\mathsf{attack} = \langle \mathsf{active}, t \in [t_4, t_4 + 1) \rangle)$ mined from device logs from an enterprise network. It predicts an active attack on subnet x_0 within the next one min, if the following event series have been reported where consecutive events are at most one min apart from each other: (1) an IPS device detects an Nmap event (x_1); (2) a WAF detects a scanner operation (x_2); (3) an IPS device reports a Startracker alert (x_3); and (4) an IPS device catches a WannaRen transmission (x_4). This rule has been deployed at a large company, where its accuracy is over 85%. Better still, it allows preemptive defensive measures *before* the attacks on the subnet are carried out.

4 Enriching and Cleaning Graphs

Fishing Fort is also able to enrich (Section 4.1) and clean (Section 4.2) real-life graphs.

4.1 Graph Enrichment

It is common to find the data in a real-life graph G_1 "incomplete", witnessed by missing properties and links. This is because in contrast to relational databases, graph G_1 is often "schemaless"; in such a graph, entities are specified by vertices and their subgraphs with irregular structures in the absence of constraints. With this comes the need for filling in the information missing from G_1 with data from external graphs G_2 that have overlapping information, thereby improving the accuracy of association analyses in graph G_1.

The need for referencing external graphs has been recognized in medical knowledge discovery [38], which aims to find semantic patterns and improve the accuracy and efficiency of diagnoses and treatment. As reported in [38, 45], as high as 77% of the discovered patterns span across heterogeneous biological networks. Moreover, e-commerce requires inspecting multiple graphs since an average social media user engages 6.6 different platforms [7]. Analyzing a single platform cannot fully understand users' interests. In addition, recommendation [41], information diffusing prediction [51] and network dynamics analysis [49] have been deducing associations across multiple graphs. Fishing Fort also enriches data for, e.g., drug discovery.

Fishing Fort implements the data enrichment algorithm of [19]. Given a real-life graph G_1 and an external graph G_2 with overlapping information, it enriches G_1 with only relevant information from G_2, to reduce noise and cost. It develops a graph filtering method to (a) identify vertices u in G_1 and v in G_2 that refer to the same real-world entity via heterogeneous entity resolution [15], and (b) locate relevant data of v in G_2 that pertains to u in G_1 by training an ML model. It supports two modes: (1) a physical mode by extracting properties of v from G_2 and enriches u by adding the data to G_1; and (2) a virtual mode by discovering GARs that pertain to entities in G_1 across G_1 and G_2, without physically merging the two; the discovered GARs are applied to G_1 and the filtered subgraphs of G_2, again without merging the two. Fishing Fort supports batch and incremental discovery of such GARs.

For drug discovery, Fishing Fort builds a uniform drug-disease graph by integrating data from eleven libraries and data banks, including CTD [5], MeSH [35] and BioGrid [2]. Given data banks $G_1 = (V_1, E_1, L_1, F_1)$ and $G_2 = (V_2, E_2, L_2, F_2)$, Fishing Fort "joins" the two as $G_\oplus(G_1, G_2) = (V_\oplus, E_\oplus, L_\oplus, F_\oplus)$, where $V_\oplus$ (resp. $E_\oplus$) is a revision of the union $V_1 \cup V_2$ (resp. $E \cup E_2$) such that u and v are represented as the same (merged) vertex in $V_\oplus$ if (u, v) is a match by parametric simulation [15]; and $L_\oplus$ and $F_\oplus$ inherit the label and attribute assignments from G_1 and G_2. When u and v both carry attribute A, the merged vertex takes the value $v.A$ from G_i ($\in [1, 2]$) if the data in G_i is more reliable. It incrementally enriches the graph in the physical mode by extracting data from external sources [19]. Moreover, it cleans the graph by detecting and fixing duplicates and inconsistencies (see Section 4.2).

4.2 Graph cleaning

Critical to graph analytics is the quality of the data in graphs. Real-life graphs often have duplicates and conflicts. Even knowledge graphs widely in use include such errors. In Yago, for instance, two different vertices named "Motorola" and "Team Motorola" refer to the same cycling team sponsored by Motorola, yielding a duplicate. The Baron Hirsch Synagogue, a building designed by architect George Awsumb, is labeled as a "flagship" of American Orthodox Judaism, while in DBpedia, this synagogue is classified as a ship. It is known that 26% of triples in knowledge graph NELL bear conflicts [52], and duplicate entities of Wikidata are involved in 27.8% factual statements it provides [39]. Moreover, noise in biomedical knowledge graphs is considered a big challenge to drug discovery [50]. Thus, two primitive issues of graph data quality are (a) *entity resolution (ER)*, to identify vertices that refer to the same real-world entity, and (b) *conflict resolution (CR)*, to fix inconsistencies among entities. The situation gets worse when it comes to data merged from multiple sources.

Besides ER and CR, there are two other critical issues of data quality. One is *timeliness* (a.k.a. data currency), for how up-to-date the information is. The other is *information completeness*, for the availability of data on-hand for analytics. In the real world, data easily becomes obsolete, and stale data is inaccurate. For instance, "82% of companies are making decisions based on stale information", and 85% of the companies witness that "this stale data is leading to incorrect decisions and lost revenue" [3]. Equally damaging is missing data. As revealed by a poll on clinical data management challenges in 2018, 77% of the participants rated missing patient data as a critical problem [42], which may result in incorrect diagnoses.

Hence the need is evident for (c) *timeliness deduction* (TD), to deduce temporal orders on attribute values, and (d) *missing data imputation* (MI), to fill in missing values/links.

One could use GARs to clean graphs. Indeed, GARs can express rules for ER, CR and MI. However, as indicated in Section 2.3, it is intractable to detect and fix errors with GARs. Moreover, GARs stop short of supporting temporal ranking and deducing timeliness.

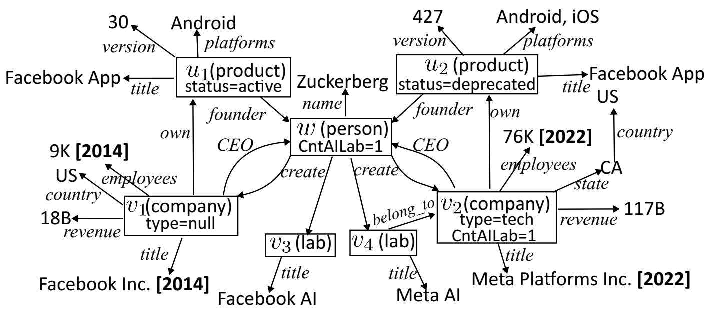


Figure 3 A graph G of companies and products.

Graph Cleaning Rules. In light of these, Fishing Fort develops a class of rules, referred to as GCRs [12]. A GCR has the form of $\mathcal{Q}[x_0, y_0](X \to p_0)$. It differs from a GAR in the following.

(1) On the one hand, it adopts a restricted form of graph patterns: $\mathcal{Q}[x_0, y_0]$ is a pair $\langle Q_x[x_0, \bar{x}], Q_y[y_0, \bar{y}]\rangle$ of patterns, where Q_x has a "star" shape centered at a vertex x_0, which denotes an entity of interest, and links to a set of characteristic features (leaves); similarly for Q_y. Intuitively, $\mathcal{Q}[x_0, y_0]$ represents two entities x_0 and y_0 with (possibly) different types and heterogeneous structures in a schemaless graph. It specifies features for pairwise comparison between x_0 and y_0. GCRs employ dual star-shaped patterns rather than general (possibly cyclic) patterns in GARs. It is in PTIME to check the matches of such patterns in a graph, as opposed to the intractability of general patterns.

(2) On the other hand, in addition to all the predicates of GARs, GCRs support temporal predicates $x.A \prec y.B$ (resp. $x.A \preceq y.B$), indicating that attribute $y.B$ is more current than $x.A$ (resp. at least as current as $x.A$). Moreover, Fishing Fort trains a temporal ranking model $\mathcal{M}_{\mathsf{rank}}$ $(x.A, y.B)$, which returns true iff $\mathcal{M}_{\mathsf{rank}}$ predicts that $x.A \preceq y.B$.

(3) GCRs further restrict the preconditions X such that all binary predicates are defined on two leaves x and y in Q_x and Q_y of $\mathcal{Q}$, respectively, and each leaf may carry at most one such predicate (but multiple unary predicates $z.A \oplus c$).

GCRs are able to express rules for ER, CR, TD and MI. As an example, DBpedia and Yago include two company entities having distinct titles "Facebook, Inc." and "Meta Platforms, Inc.", without any timestamp. Worse yet, other data of these entities is either outdated or missing, e.g., revenue, research labs, and type. There are also two products for the online social networking service named "Facebook App". They carry different values for, e.g., current status, version numbers and platforms, but with few timestamps. In fact, snapshot-based knowledge graphs, e.g., DBpedia, maintain the "most recent" facts with no timestamps. Figure 3 depicts such a graph G that includes the product entities u_1, u_2, and companies v_1, v_2. To catch errors, we may use the GCRs below with dual patterns in Figure 4.

(1) $\phi_1 = \mathcal{Q}_1[x_0, y_0](X_1 \wedge x_3.\mathsf{val} < y_3.\mathsf{val} \to x_3.\mathsf{val} \prec y_3.\mathsf{val})$, where X_1 is $x_1.\mathsf{id} = y_1.\mathsf{id} \wedge x_2.\mathsf{val} = y_2.\mathsf{val} \wedge \mathcal{M}_{\mathsf{rank}}(x_4.\mathsf{val}, y_4.\mathsf{val})$. It deduces temporal orders from the version numbers of an online service App (product), i.e., if two services x_0 and y_0 are created by the same person, have the same title, and if model $\mathcal{M}_{\mathsf{rank}}$ predicts that y_0 has newer value for platforms (specified in X_1), then the larger version number of y_0 is more current (for TD). In practice, the version number of an App grows as the platforms evolve.

(2) $\phi_2 = \mathcal{Q}_1[x_0, y_0](X_1 \wedge x_3.\mathsf{val} \prec y_3.\mathsf{val} \wedge x_0.\mathsf{status} = \mathsf{active} \to y_0.\mathsf{status} = \mathsf{active})$. That is, if service y_0 has newer values for version numbers (by $x_3.\mathsf{val} \prec y_3.\mathsf{val}$) than x_0 and if x_0 is active, then with the same condition X_1 as in (1), y_0 should also be active (for CR).

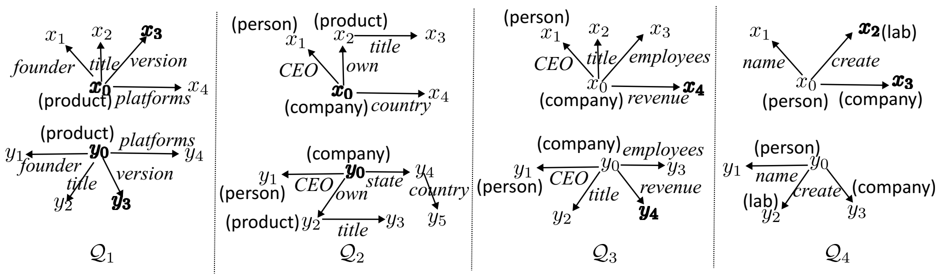


Figure 4 Example dual patterns.

(3) $\phi_3 = Q_2[x_0, y_0](x_1.\mathsf{id} = y_1.\mathsf{id} \wedge x_2.\mathsf{status} = \text{active} \wedge y_2.\mathsf{status} = \text{active} \wedge x_3.\mathsf{val} = y_3.\mathsf{val} \wedge x_4.\mathsf{val} = y_5.\mathsf{val} \rightarrow x_0.\mathsf{id} = y_0.\mathsf{id})$. It identifies two companies x_0 and y_0 (for ER), if they have the same CEO and country of location, and if they own *active* products of the same title. Here the country property is fetched along different paths in the two stars of Q_2.

(4) $\phi_4 = Q_3[x_0, y_0](x_0.\mathsf{type} = \text{tech} \wedge y_0.\mathsf{type} = \text{tech} \wedge x_1.\mathsf{id} = y_1.\mathsf{id} \wedge \mathcal{M}_s(x_2.\mathsf{val}, y_2.\mathsf{val}) \wedge x_3.\mathsf{val} \prec y_3.\mathsf{val} \rightarrow x_4.\mathsf{val} \prec y_4.\mathsf{val})$. It deduces that company y_0 has a newer revenue than x_0 (for TD) if the number of employees of y_0 is more current, and both x_0 and y_0 are tech companies with the same CEO and similar titles (checked by ML model $\mathcal{M}_s$).

(5) $\phi_5 = Q_4[x_0, y_0](x_1.\mathsf{val} = y_1.\mathsf{val} \wedge \mathcal{M}_e(x_2, y_2) \wedge x_3.\mathsf{id} = y_3.\mathsf{id} \wedge y_0.\mathsf{CntAILab} = 1 \wedge y_3.\mathsf{CntAILab} = 1 \rightarrow \mathsf{belong\_to}(x_2, x_3))$. The GCR states that if companies x_3, y_3 and AI labs x_2, y_2 are created by persons x_0 and y_0 of the same name, x_3 and y_3 are the same entity, x_2 and y_2 are identified by an ER model $\mathcal{M}_e$, and if y_0 and y_3 pertain to AI lab y_2, then x_2 is an AI lab belonging to x_3. It adds a missing link from x_2 to x_3 (for MI).

Moreover, Fishing Fort leverages the interaction among ER, CR, TD, and MI to improve the overall quality of the graph. Consider the GCRs above and graph G of Figure 3.

(a) Initially, we deduce that product u_2's version number (427) is more current than the version (30) of u_1 by using GCR ϕ_1 (TD).

(b) Then a conflict between the status values of products u_1 and u_2 is found by GCR ϕ_2, and u_2's status can be rectified to be active (CR).

(c) We next identify companies v_1 and v_2 by using GCR ϕ_3 (ER), where pattern nodes x_4 and y_5 in Q_2 are mapped to vertices with value US.

(d) The type at v_1 is then copied from (more reliable) v_2 (MI), as a byproduct of ER.

(e) Next the revenue of v_2 is verified to be more current than v_1, via GCR ϕ_4 (TD).

(f) Finally, we deduce a missing link by GCR ϕ_5, i.e., the AI lab v_3 belongs to v_1 (MI).

ER, CR, TD and MI help each other. Indeed, ER step (c), CR step (b), TD steps (a) and (e), and MI steps (d) and (f) interact with each other in this process. For instance, step (b) is applicable only after we deduce temporal orders for version numbers in (a); and step (e) needs the results of ER and MI in (c)-(d), which decide the right type value for company v_1.

Complexity. The restrictions on graph patterns and preconditions make it easier to clean graphs with GCRs. The validation problem becomes tractable for GCRs. Moreover, error detection and correction with GCRs are also in PTIME (see below). One step further, the satisfiability and implication problems are in PTIME if we consider GCRs with predicates $x.A \oplus y.B$, $x.A \oplus c$ and $\mathcal{M}(x.\bar{A}, y.\bar{B})$ only, where $\oplus$ is either $=$ or $\neq$ [12]. These problems become intractable under any of the following conditions, unless P = NP: (a) in the presence of link predicates, temporal predicates or comparison $\leq, <, \geq, >$, which are necessary for, e.g., CR, TD, and MI; or (b) if the restrictions on the preconditions X of GCRs are lifted.

Algorithms. Given an (enriched) real-life graph G, Fishing Fort discovers a set Σ of GCRs and detects and fixes errors with the GCRs using the following algorithms.

(a) *Rule discovery* [12]. Fishing Fort discovers GCRs from samples of G as remarked earlier [13]. In contrast to how it mines GARs, it makes practical use of star patterns to speed up the discovery process [12]. It computes "scattered matches" of a star pattern, i.e., the matches $(h(x_0), h(y_0))$ of the center nodes in $\mathcal{Q}[x_0, y_0]$, rather than complete homomorphic matches h of $\mathcal{Q}$. The scattered matches can be computed in PTIME for star patterns via dynamic programming, as opposed to (possibly) exponential time for complete matches of general patterns. As shown in [12], it is much faster to discover GCRs than GARs.

(b) *Error detection* [12]. Employing the discovered GCRs in Σ, Fishing Fort detects errors in G, i.e., violations of a GCR $\phi = \mathcal{Q}[x_0, y_0](X \to p_0)$ in Σ, which are scattered matches $(h(x_0), h(y_0))$ for a match h such that $h \models X$ but $h \not\models p_0$. The violations can be duplicates, conflicts, conflicting timeline or missing links/attributes.

(c) *Error correction.* Fishing Fort enforces GCRs in Σ on graph G via chase [21], to deduce fixes. Besides ER and CR, in the process it determines temporal orders on graph properties, and fills in missing values and links by possibly referencing ground truth Γ and knowledge graph KG. It updates G with the deduced fixes, e.g., new edges for missing links. Moreover, it adds the fixes to ground truth Γ for subsequent error corrections. The chase is also Church-Rosser, and the fixes are logical consequences of Σ and Γ, i.e., the fixes are validated as long as the GCRs and ground truth are correct. In the process, it suffices to compute scattered matches of patterns, rather than complete ones.

All these algorithms are parallelly scalable and can scale with large graphs in principle. Moreover, the error detection and correction algorithms are in PTIME via scattered matches.

5 Conclusion

The novelty of Fishing Fort consists of the following. (1) It makes a first effort to conduct ML prediction and logic deduction in the same process, by embedding ML models as predicates in logic rules. As verified by real-life applications, the unification improves the accuracy of association analyses and the explainability of ML predictions. (2) Fishing Fort automatically discovers rules from real-life graphs, and it supports not only association analyses but also data enrichment and cleaning. (3) It can be readily adapted to different domains and has proven effective in a variety of applications. (4) The algorithms of Fishing Fort are provably parallelly scalable and hence in principle, can scale with large graphs.

We are currently adapting Fishing Fort to quantitative trading, fraud detection, and manufacturing industry beyond EV batteries. The preliminary results are quite encouraging.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

2 BioGRID, 2024. URL: `https://thebiogrid.org/`.

3 Businesswire. Over 80 percent of companies rely on stale data for decision-making, 2022. *https://www.businesswire.com/news/home/20220511005403/en/Over-80-Percent-of-Companies-Rely-on-Stale-Data-for-Decision-Making*.

4 Jin-yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identifications. *Comb.*, 12(4):389–410, 1992.

5 Comparative Toxicogenomics Database (CTD), 2024. URL: `https://ctdbase.org/`.

6 A Dairam, Edith M Antunes, KS Saravanan, and Santylal Daya. Non-steroidal anti-inflammatory agents, tolmetin and sulindac, inhibit liver tryptophan 2, 3-dioxygenase activity and alter brain neurotransmitter levels. *Life sciences*, 79(24):2269–2274, 2006.

7 Brian Dean. Social network usage and growth statistics. *https://backlinko.com/social-media-users*, 2023.

8 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

9 Exasol. Exasol research finds 58% of organizations make decisions based on outdated data, 2020. *https://www.exasol.com/news-exasol-research-finds-organizations-make-decisions-based-on-outdated-data/*.

10 Lihang Fan, Wenfei Fan, Ping Lu, Chao Tian, and Qiang Yin. Enriching recommendation models with logic conditions. *Proc. ACM Manag. Data*, 2024.

11 Wenfei Fan. Big graphs: Challenges and opportunities. *PVLDB*, 15(12):3782–3797, 2022.

12 Wenfei Fan, Wenzhi Fu, Ruochun Jin, Muyang Liu, Ping Lu, and Chao Tian. Making it tractable to catch duplicates and conflicts in graphs. *Proc. ACM Manag. Data*, 1(1):86:1–86:28, 2023.

13 Wenfei Fan, Wenzhi Fu, Ruochun Jin, Ping Lu, and Chao Tian. Discovering association rules from big graphs. *PVLDB*, 15(7):1479–1492, 2022.

14 Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. on Database Systems*, 33(1), 2008.

15 Wenfei Fan, Liang Geng, Ruochun Jin, Ping Lu, Resul Tugey, and Wenyuan Yu. Linking entities across relations and graphs. In *ICDE*, pages 634–647. IEEE, 2022.

16 Wenfei Fan, Tao He, Longbin Lai, Xue Li, Yong Li, Zhao Li, Zhengping Qian, Chao Tian, Lei Wang, Jingbo Xu, Youyang Yao, Qiang Yin, Wenyuan Yu, Kai Zeng, Kun Zhao, Jingren Zhou, Diwen Zhu, and Rong Zhu. GraphScope: A unified engine for big graph processing. *PVLDB*, 14(12):2879–2892, 2021.

17 Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Chao Tian, and Jingren Zhou. Capturing associations in graphs. *PVLDB*, 13(11):1863–1876, 2020.

18 Wenfei Fan, Ruochun Jin, Ping Lu, Chao Tian, and Ruiqi Xu. Towards event prediction in temporal graphs. *PVLDB*, 15(9):1861–1874, 2022.

19 Wenfei Fan, Muyang Liu, Shuhao Liu, and Chao Tian. Capturing more associations by referencing knowledge graphs. *PVLDB*, 2024.

20 Wenfei Fan and Ping Lu. Dependencies for graphs. *ACM Trans. Database Syst.*, 44(2):5:1–5:40, 2019.

21 Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. Deducing certain fixes to graphs. *PVLDB*, 12(7):752–765, 2019.

22 Wenfei Fan, Xin Wang, Yinghui Wu, and Jingbo Xu. Association rules with graph patterns. *PVLDB*, 8(12):1502–1513, 2015.

23 Wenfei Fan, Yinghui Wu, and Jingbo Xu. Functional dependencies for graphs. In *SIGMOD*, pages 1843–1857. ACM, 2016.

24 Wenfei Fan, Wenyuan Yu, Jingbo Xu, Jingren Zhou, Xiaojian Luo, Qiang Yin, Ping Lu, Yang Cao, and Ruiqi Xu. Parallelizing sequential graph computations. *ACM Trans. Database Syst.*, 43(4):18:1–18:39, 2018.

25 Chris Fotis, Asier Antoranz, Dimitris Hatziavramidis, Theodore Sakellaropoulos, and Leonidas G. Alexopoulos. Pathway-based technologies for early drug discovery. *Drug Discovery Today*, 2017.

26 Martin Grohe. word2vec, node2vec, graph2vec, x2vec: Towards a theory of vector embeddings of structured data. In *PODS*, pages 1–16. ACM, 2020.

27 Yang Hu, Xiyuan Wang, Zhouchen Lin, Pan Li, and Muhan Zhang. Two-dimensional Weisfeiler-Lehman graph neural networks for link prediction. *CoRR*, abs/2206.09567, 2022.

28 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*, 2017.

29 Clyde P. Kruskal, Larry Rudolph, and Marc Snir. A complexity theory of efficient parallel algorithms. *Theor. Comput. Sci.*, 71(1):95–132, 1990.

30 Market Research Intellignece Lab. Battery formation and grading system market size, outlook: Share, growth, and forecast (2024-2031), 2024. *https://www.linkedin.com/pulse/battery-formation-grading-system-market-yae8f/*.

31 Ying Lai, Giorgio Fois, Jose R Flores, Michael J Tuvim, Qiangjun Zhou, Kailu Yang, Jeremy Leitz, John Peters, Yunxiang Zhang, Richard A Pfuetzner, Luis Esquivies, Philip Jones, Manfred Frick, Burton F. Dickey, and Axel T. Brunger. Inhibition of calcium-triggered secretion by hydrocarbon-stapled peptides. *Nature*, 603(7903):949–956, 2022.

32 Jeanne C Latourelle, Merete Dybdahl, Anita L Destefano, Richard H Myers, and Timothy L Lash. Risk of parkinson's disease after tamoxifen treatment. *BMC neurology*, 10(1):1–7, 2010.

33 Bing Li, Wei Wang, Yifang Sun, Linhan Zhang, Muhammad Asif Ali, and Yi Wang. GraphER: Token-centric entity resolution with graph convolutional neural networks. In *AAAI*, pages 8172–8179, 2020.

34 Yu Li, Hiroyuki Kuwahara, Peng Yang, Le Song, and Xin Gao. PGCN: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks. *biorxiv*, page 532226, 2019.

35 Medical Subject Headings (MeSH), 2024. URL: `https://www.nlm.nih.gov/mesh/`.

36 R Sandyk and MA Gillman. Acute exacerbation of parkinson's disease with sulindac. *Annals of neurology*, 17(1):104–105, 1985.

37 Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web (ESWC)*, pages 593–607. Springer, 2018.

38 Feichen Shen and Yugyung Lee. Knowledge discovery from biomedical ontologies in cross domains. *PloS one*, 11(8):e0160005, 2016.

39 Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro A. Szekely. A study of the quality of Wikidata. *J. Web Semant.*, 72:100679, 2022.

40 Juan Shu, Yu Li, Sheng Wang, Bowei Xi, and Jianzhu Ma. Disease gene prediction with privileged information and heteroscedastic dropout. *Bioinformatics*, 37(Supplement_1):i410–i417, 2021.

41 Kai Shu, Suhang Wang, Jiliang Tang, Reza Zafarani, and Huan Liu. User identity linkage across online social networks: A review. *SIGKDD Explor.*, 18(2):5–17, 2016.

42 Julie Smiley. Missing data and its impact on clinical research, 2016. *https://blogs.oracle.com/health-sciences/post/missing-data-and-its-impact-on-clinical-research*.

43 Bo-Tao Tan, Li Wang, Sen Li, Zai-Yun Long, Ya-Min Wu, and Yuan Liu. Retinoic acid induced the differentiation of neural stem cells from embryonic spinal cord into functional neurons in vitro. *International journal of clinical and experimental pathology*, 8(7), 2015.

44 Xiaochan Wang, Yuchong Gong, Jing Yi, and Wen Zhang. Predicting gene-disease associations from the heterogeneous network using graph embedding. In *IEEE International conference on bioinformatics and biomedicine (BIBM)*, pages 504–511, 2019.

45 Antony J Williams, Lee Harland, Paul Groth, Stephen Pettifer, Christine Chichester, Egon L Willighagen, Chris T Evelo, Niklas Blomberg, Gerhard Ecker, Carole Goble, and Barend Mons. Open PHACTS: semantic interoperability for drug discovery. *Drug discovery today*, 17(21-22):1188–1198, 2012.

46 Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*, pages 9240–9251, 2019.

47 Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *ICML*, pages 12241–12252. PMLR, 2021.

48 Yuebo Yuan, Xiangdong Kong, Jianfeng Hua, Yue Pan, Yukun Sun, Xuebing Han, Hongxin Yang, Yihui Li, Xiaoan Liu, Xiaoyi Zhou, Languang Lu, and Hewu Wang. Fast grading method based on data driven capacity prediction for high-efficient lithium-ion battery manufacturing. *Journal of Energy Storage*, 73:109143, 2023.

49 Reza Zafarani and Huan Liu. Users joining multiple sites: Friendship and popularity variations across sites. *Inf. Fusion*, 28:83–89, 2016.

50 Xiangxiang Zeng, Xinqi Tu, Yuansheng Liu, Xiangzheng Fu, and Yansen Su. Toward better drug discovery with knowledge graph. *Current opinion in structural biology*, 72:114–126, 2022.

51 Qianyi Zhan, Jiawei Zhang, Senzhang Wang, Philip S. Yu, and Junyuan Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD*, pages 58–69, 2015.

52 Qinggang Zhang, Junnan Dong, Keyu Duan, Xiao Huang, Yezi Liu, and Linchuan Xu. Contrastive knowledge graph error detection. In *CIKM*, 2022.

53 Jie Zhao, Manish Kumar, Jeevan Sharma, and Zhihai Yuan. Arbutin effectively ameliorates the symptoms of parkinson's disease: The role of adenosine receptors and cyclic adenosine monophosphate. *Neural regeneration research*, 16(10):2030, 2021.

A Note on Logical PERs and Reducibility. Logical Relations Strike Again!

Jean Gallier
University of Pennsylvania, Philadelphia, PA, USA

Abstract

We prove a general theorem for establishing properties expressed by binary relations on typed (first-order) λ-terms, using a variant of the reducibility method and logical PERs. As an application, we prove simultaneously that β-reduction in the simply-typed λ-calculus is strongly normalizing, and that the Church-Rosser property holds (and similarly for $\beta\eta$-reduction).

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Type theory; Theory of computation → Complexity theory and logic

Keywords and phrases Typed lambda-calculus, reducibility, logical relations, per's, strong normalization, Church-Rosser property

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.7

1 Introduction

Logical relations are an important tool used in proving some deep results about various typed λ-calculi and their models. A special form of the concept of a logical relation first appeared in Harvey Friedman's seminal paper [4]. General logical relations were defined and used extensively in the pioneering work of Plotkin [18] and Statman [19, 21, 20], and later on in a more general setting by Breazu-Tannen and Coquand [2], Mitchell [15], Mitchell and Moggi [16], and Abramsky [1], among others. As the name indicates, logical relations are certain kinds of relations, and they are used to prove relational properties of terms. On the other hand, reducibility is a tool used in proving properties of terms in various typed λ-calculi. Typically, it is used to prove strong normalization or normalization, but it can be used to prove other properties as well. The method was pioneered by Tait [22] for the simply-typed λ-calculus, and brilliantly extended to various higher-order typed λ-calculi by Girard [9, 10] (see also Tait [23]). Various expositions and analyses of such proofs are given in Mitchell [15], Krivine [14], Huet [11], and Gallier [5, 6, 7, 8], among others. Another crucial concept is that of a partial equivalence relation, or *PER*. PER's were introduced by Hyland [12] and Mulry [17]. PERs are a major tool in defining categories of domains in an effective setting (see Freyd, Mulry, Rosolini, and Scott [3]). PERs also often show up as logical relations, and are called *logical PERs* (see Breazu-Tannen and Coquand [2]).

In this note, we prove a general theorem for establishing properties expressed by binary relations on typed (first-order) λ-terms, using a variant of the reducibility method and of logical PERs. This note is written much in the spirit of our earlier papers [6, 8]. Our goal is to elucidate the conditions under which the technology of reducibility and of logical relations works. We do this by finding sufficient conditions that a binary relation $\mathcal{R}$ on typed λ-terms need to satisfy for establishing that $\mathcal{R}$ holds, using reducibility. The conditions presented in this paper were inspired by a paper by Koletsos [13].

In this short note, we restrict out attention to the simply-typed λ-calculus, but there is little doubt that our method can be generalized to all the first-order types (as in [6]), or to type intersection disciplines (as in [8]). As an illustration, it is easy to show simultaneously that β-reduction is strongly normalizing and that the Church-Rosser property holds (and similarly for $\beta\eta$-reduction).

The generalization to second-order types (or more general types) is much more problematic (for a discussion of some of the problems, see Breazu-Tannen and Coquand [2]), and is left as an open problem.

2 $\mathcal{R}$-Logical Candidates for the Arrow Type Constructor $\to$

Let $\mathcal{T}$ denote the set of (simple) types. Recall that the set of simple types is defined inductively from a set of base types and using the type constructor $\to$, i.e. a base type b is a type, and $(\sigma \to \tau)$ is a type whenever σ and τ are types.

The presentation will be simplified if we adopt the definition of simply-typed λ-terms where all the variables are explicitly assigned types once and for all. More precisely, we have a family $\mathcal{X} = (X_\sigma)_{\sigma\in\mathcal{T}}$ of variables, where each X_σ is a countably infinite set of variables of type σ, and $X_\sigma \cap X_\tau = \emptyset$ whenever $\sigma \neq \tau$. Using this definition, there is no need to drag contexts along, and the most important feature of the proof, namely the reducibility method, is easier to grasp. Recall that an untyped λ-term is either a variable x, an application (MN), or a λ-abstraction $\lambda x\colon \sigma.\, M$. The terms of the typed λ-calculus $\lambda^{\to}$ (also called simply-typed λ-terms) are the λ-terms that respect certain type-checking rules reviewed below.

▶ **Definition 1.** *Given a λ-term M and a type σ, we define the binary relation $M\colon \sigma$ (read, M has type σ) using the following type-checking rules:*

$$x\colon \sigma, \quad \textit{when } x \in X_\sigma,$$

(we can also have $c\colon \sigma$, where c is a constant of type σ, if there is a set of constants that have been preassigned types).

$$\frac{x\colon \sigma \quad M\colon \tau}{\lambda x\colon \sigma.\, M\colon (\sigma \to \tau)} \quad \textit{(abstraction)}$$

$$\frac{M\colon (\sigma \to \tau) \quad N\colon \sigma}{(MN)\colon \tau} \quad \textit{(application)}$$

From now on, when we refer to a λ-term, we mean a λ-term that type-checks. We let Λ_σ denote the set of λ-terms of type σ, and $\Lambda^{\to} = (\Lambda_\sigma)_{\sigma\in\mathcal{T}}$, also called the set of *simply-typed λ-terms*. In this section, the only reduction rule considered is β-reduction:

$$(\lambda x\colon \sigma.\, M)N \longrightarrow_\beta M[N/x].$$

Equations between λ-terms of the same type σ are denoted as $M \doteq N\colon \sigma$, and equational provability is defined as follows.

▶ **Definition 2.** *The axioms and inference rules of the equational β-theory of the typed λ-calculus $\lambda^{\to}$ are defined below.*

$$x \doteq x\colon \sigma \quad \textit{(reflexivity)},$$

where x is any variable of type σ. We also have axioms $c \doteq c\colon \sigma$, where c is a constant of type σ, when typed constants are present.

$$(\lambda x\colon \sigma.\, M)N \doteq M[N/x]\colon \tau \quad (\beta)$$

$$\frac{M_1 \doteq M_2 \colon \sigma}{M_2 \doteq M_1 \colon \sigma} \quad \textit{(symmetry)}$$

$$\frac{M_1 \doteq M_2 \colon \sigma \quad M_2 \doteq M_3 \colon \sigma}{M_1 \doteq M_3 \colon \sigma} \quad \textit{(transitivity)}$$

$$\frac{M_1 \doteq M_2 \colon (\sigma \to \tau) \quad N_1 \doteq N_2 \colon \sigma}{(M_1 N_1) \doteq (M_2 N_2) \colon \tau} \quad \textit{(congruence)}$$

$$\frac{M_1 \doteq M_2 \colon \tau}{\lambda x \colon \sigma.\, M_1 \doteq \lambda x \colon \sigma.\, M_2 \colon (\sigma \to \tau)} \quad (\xi)$$

The notation $\vdash_\beta M \doteq N \colon \sigma$ *means that the equation* $M \doteq N \colon \sigma$ *is provable from the above axioms and inference rules.*

The equational $\beta\eta$*-theory of the typed* λ*-calculus* $\lambda^{\to}$ *is obtained by adding the following axiom to the above axioms and inference rules.*

$$\lambda x \colon \sigma.\, (Mx) \doteq M \colon (\sigma \to \tau) \quad (\eta)$$

where $x \notin FV(M)$.

The notation $\vdash_{\beta\eta} M \doteq N \colon \sigma$ *means that the equation* $M \doteq N \colon \sigma$ *is provable from all the axioms, including* (η)*, and the inference rules.*

Given any term M, we can easily show by induction on the structure of M that the equation $M \doteq M \colon \sigma$ is provable using the (*reflexivity*) axioms and the rules (*congruence*) and (ξ). Thus, reflexivity holds for all terms, not just variables and constants. The reason for using a restricted form of the reflexivity axioms is that this makes the proof of Lemma 10 simpler.

It turns out that the behavior of a term depends heavily on the nature of the last typing inference rule used in typing this term. A term created by an introduction rule, or I-term, plays a crucial role, because when combined with another term, a new redex is created. On the other hand, for a term created by an elimination rule, or simple term, no new redex is created when this term is combined with another term. This motivates the following definition.

▶ **Definition 3.** *An I-term is a term of the form* $\lambda x \colon \sigma.\, M$*. A simple term (or neutral term) is a term that is not an I-term. Thus, a simple term is either a variable* x*, a constant* c*, or an application* MN*. A term* M *is stubborn iff it is simple and, either* M *is irreducible, or* M' *is a simple term whenever* $M \xrightarrow{+}_\beta M'$ *(equivalently,* M' *is* **not** *an I-term).*

Let $\mathcal{R} = (R_\sigma)_{\sigma \in \mathcal{T}}$ be a family of nonempty binary relations, where $R_\sigma \subseteq \Lambda_\sigma \times \Lambda_\sigma$.

▶ **Definition 4.** *Properties (P0)-(P3) are defined as follows:*
(P0) *Every relation* R_σ *is a per, i.e.,* R_σ *is symmetric and transitive.*
(P1) $\langle x,\, x\rangle \in R_\sigma$, $\langle c,\, c\rangle \in R_\sigma$*, for every variable* x *and constant* c *of type* σ*.*
(P2) *If* $\langle M_1,\, M_2\rangle \in R_\sigma$ *and* $M_1 \longrightarrow_\beta M_1'$*, then* $\langle M_1',\, M_2\rangle \in R_\sigma$*.*
(P3) *If* M_1 *and* M_2 *are simple,* $\langle M_1,\, M_2\rangle \in R_{\sigma\to\tau}$*,* $\langle N_1,\, N_2\rangle \in R_\sigma$*, and either* $\langle(\lambda x \colon \sigma.\, M_1')N_1,\, M_2N_2\rangle \in R_\tau$ *whenever* $M_1 \xrightarrow{+}_\beta \lambda x \colon \sigma.\, M_1'$ *and* M_2 *is stubborn, or* $\langle(\lambda x \colon \sigma.\, M_1')N_1,\, (\lambda x \colon \sigma.\, M_2')N_2\rangle \in R_\tau$ *whenever* $M_1 \xrightarrow{+}_\beta \lambda x \colon \sigma.\, M_1'$ *and* $M_2 \xrightarrow{+}_\beta \lambda x \colon \sigma.\, M_2'$*, then* $\langle M_1N_1,\, M_2N_2\rangle \in R_\tau$*.*

From now on, we only consider families of relations $\mathcal{R}$ satisfying conditions (P0)-(P3) of Definition 4.

▶ **Definition 5.** *For any type σ, a nonempty relation $C \subseteq \Lambda_\sigma \times \Lambda_\sigma$ is a $\mathcal{R}$-logical candidate iff it satisfies the following conditions:*

(R0) *C is a per.*

(R1) *$C \subseteq R_\sigma$.*

(R2) *If $\langle M_1,\ M_2\rangle \in C$ and $M_1 \longrightarrow_\beta M_1'$, then $\langle M_1',\ M_2\rangle \in C$.*

(R3) *If M_1 and M_2 are simple, $\langle M_1,\ M_2\rangle \in R_\sigma$, and either $\langle \lambda x\colon \gamma.\, M_1',\ M_2\rangle \in C$ whenever $M_1 \xrightarrow{+}_\beta \lambda x\colon \gamma.\, M_1'$ and M_2 is stubborn, or $\langle \lambda x\colon \gamma.\, M_1',\ \lambda x\colon \gamma.\, M_2'\rangle \in C$ whenever $M_1 \xrightarrow{+}_\beta \lambda x\colon \gamma.\, M_1'$ and $M_2 \xrightarrow{+}_\beta \lambda x\colon \gamma.\, M_2'$, then $\langle M_1,\ M_2\rangle \in C$.*

Note that (R3) and (P1) imply that for every type σ, any $\mathcal{R}$-logical candidate C of type σ contains all pairs $\langle x,\ x\rangle$ and $\langle c,\ c\rangle$ for all variables and all constants of type σ. More generally, (R3) implies that C contains all pairs $\langle M_1,\ M_2\rangle$ of stubborn terms in R_σ, and (P1) guarantees that pairs $\langle x,\ x\rangle$ and $\langle c,\ c\rangle$ are in R_σ (for every type σ).

By (P3), if $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$ is a pair of stubborn terms and $\langle N_1,\ N_2\rangle \in R_\sigma$ is any pair of terms, then $\langle M_1N_1,\ M_2N_2\rangle \in R_\tau$. Furthermore, M_1N_1 and M_2N_2 are also stubborn since they are simple terms and since they can only reduce to an I-term (a λ-abstraction) if M_1 or M_2 reduce to a λ-abstraction, i.e. an I-term. Thus, if $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$ is a pair of stubborn terms and $\langle N_1,\ N_2\rangle \in R_\sigma$ is any pair of terms, then $\langle M_1N_1,\ M_2N_2\rangle \in R_\tau$ is a pair of stubborn terms. Also, observe that if $M_1 \xrightarrow{+}_\beta M_1'$, $M_2 \xrightarrow{+}_\beta M_2'$, and $\langle M_1,\ M_2\rangle \in C$, then $\langle M_1',\ M_2'\rangle \in C$. This follows from (R2) and (R0), since (R0) implies symmetry and transitivity.

Given a family of relations $\mathcal{R}$, for every type σ, we define the relation $[\![\sigma]\!]$ as follows.

▶ **Definition 6.** *The logical relations $[\![\sigma]\!]$ are defined as follows:*

$$\begin{aligned}
[\![\sigma]\!] &= R_\sigma, \qquad \sigma \text{ a base type},\\
[\![\sigma\to\tau]\!] &= \{\langle M_1,\ M_2\rangle \mid \langle M_1,\ M_2\rangle \in R_{\sigma\to\tau},\ \text{and for all } N_1,\ N_2,\\
&\qquad \text{if } \langle N_1,\ N_2\rangle \in [\![\sigma]\!] \text{ then } \langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]\}.
\end{aligned}$$

▶ **Lemma 7.** *If $\mathcal{R}$ is a family of relations satisfying conditions (P0)-(P3), then each $[\![\sigma]\!]$ is a $\mathcal{R}$-logical candidate that contains all pairs of stubborn terms in R_σ.*

Proof. We proceed by induction on types. If σ is a base type, $[\![\sigma]\!] = R_\sigma$, and obviously, every pair of stubborn terms in R_σ is in $[\![\sigma]\!]$. Since $[\![\sigma]\!] = R_\sigma$, (R0) and (R1) are trivial, (R2) follows from (P2), and (R3) is also trivial.[1]

We now consider the induction step.

(R0). By the definition of $[\![\sigma\to\tau]\!]$, symmetry and transitivity are straightforward.

(R1). By the definition of $[\![\sigma\to\tau]\!]$, (R1) is trivial.

(R2). Let $\langle M_1,\ M_2\rangle \in [\![\sigma\to\tau]\!]$, and assume that $M_1 \longrightarrow_\beta M_1'$. Since $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$ by (R1), we have $\langle M_1',\ M_2\rangle \in R_{\sigma\to\tau}$ by (P2). For any $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, since

[1] In fact, if $[\![\sigma]\!] = R_\sigma$, (R3) holds trivially even at nonbase types. This remark is useful is we allow type variables.

$\langle M_1,\ M_2\rangle \in [\![\sigma \to \tau]\!]$, we have $\langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]$, and since $M_1 \longrightarrow_\beta M_1'$ we have $M_1N_1 \longrightarrow_\beta M_1'N_1$. Then, applying the induction hypothesis at type τ, (R2) holds for $[\![\tau]\!]$, and thus $\langle M_1'N_1,\ M_2N_2\rangle \in [\![\tau]\!]$. Thus, we have shown that $\langle M_1',\ M_2\rangle \in R_{\sigma\to\tau}$ and that if $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, then $\langle M_1'N_1,\ M_2N_2\rangle \in [\![\tau]\!]$. By the definition of $[\![\sigma \to \tau]\!]$, this shows that $\langle M_1',\ M_2\rangle \in [\![\sigma \to \tau]\!]$, and (R2) holds at type $\sigma \to \tau$.

(R3). Let $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$, and assume that $\langle \lambda x\colon \sigma.\ M_1',\ \lambda x\colon \sigma.\ M_2'\rangle \in [\![\sigma \to \tau]\!]$ whenever $M_1 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_1'$ and $M_2 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_2'$, or that $\langle \lambda x\colon \sigma.\ M_1',\ M_2\rangle \in [\![\sigma \to \tau]\!]$ whenever $M_1 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_1'$ and M_2 is stubborn, where M_1 and M_2 are simple terms. We prove that for every $\langle N_1,\ N_2\rangle$, if $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, then $\langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]$. First, we prove that $\langle M_1N_1,\ M_2N_2\rangle \in R_\tau$, and for this we use (P3). First, assume that M_1 and M_2 are stubborn, and let $\langle N_1,\ N_2\rangle$ be in $[\![\sigma]\!]$. By (R1), $\langle N_1,\ N_2\rangle \in R_\sigma$. By the induction hypothesis, all pairs of stubborn terms in R_τ are in $[\![\tau]\!]$. Since we have shown that $\langle M_1N_1,\ M_2N_2\rangle$ is a pair of stubborn terms in R_τ whenever $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$ is pair of stubborn terms and $\langle N_1,\ N_2\rangle \in R_\tau$, we have $\langle M_1,\ M_2\rangle \in [\![\sigma \to \tau]\!]$.

Now, assume that M_1 or M_2 is not stubborn. Since by (R0), each $[\![\sigma]\!]$ is symmetric, we only need to consider the case where M_1 is not stubborn and M_2 is stubborn. This case is similar to the next case, because M_2N_2 is stubborn for any N_2, and we leave it as an exercise.

Consider $\langle M_1, M_2\rangle \in R_{\sigma\to\tau}$ where M_1 and M_2 are non stubborn. If $M_1 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_1'$ and $M_2 \xrightarrow{+}_\beta \lambda x\colon \sigma. M_2'$, then by assumption, $\langle \lambda x\colon \sigma. M_1',\ \lambda x\colon \sigma. M_2'\rangle \in [\![\sigma \to \tau]\!]$, and for any $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, we have $\langle(\lambda x\colon \sigma.\ M_1')N_1,\ (\lambda x\colon \sigma.\ M_2')N_2\rangle \in [\![\tau]\!]$. Since by (R1), $\langle N_1,\ N_2\rangle \in R_\sigma$ and $\langle(\lambda x\colon \sigma.\ M_1')N_1,\ (\lambda x\colon \sigma.\ M_2')N_2\rangle \in R_\tau$, by (P3), we have $\langle M_1N_1,\ M_2N_2\rangle \in R_\tau$. Now, there are two cases.

If τ is a base type, then $[\![\tau]\!] = R_\tau$ and $\langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]$.

If τ is not a base type, then the terms M_1N_1 and M_2N_2 are simple. We prove that $\langle M_1N_1, M_2N_2\rangle \in [\![\tau]\!]$ using (R3) (which by induction, holds at type τ). The case where M_1N_1 and M_2N_2 are stubborn follows from the induction hypothesis. The case where M_1N_1 is not stubborn and M_2N_2 is stubborn is similar to the next case, but simpler (and the symmetric case follows by (R0)).

If both M_1N_1 and M_2N_2 are not stubborn terms, observe that if $M_1N_1 \xrightarrow{+}_\beta Q_1$ and $M_2N_2 \xrightarrow{+}_\beta Q_2$, where $Q_1 = \lambda y\colon \gamma.\ P_1$ and $Q_2 = \lambda y\colon \gamma.\ P_2$ are I-terms, then the reductions are necessarily of the form

$$M_1N_1 \xrightarrow{+}_\beta (\lambda x\colon \sigma.\ M_1')N_1' \longrightarrow_\beta M_1'[N_1'/x] \xrightarrow{*}_\beta Q_1,$$

and

$$M_2N_2 \xrightarrow{+}_\beta (\lambda x\colon \sigma.\ M_2')N_2' \longrightarrow_\beta M_2'[N_2'/x] \xrightarrow{*}_\beta Q_2,$$

where $M_1 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_1'$, $M_2 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_2'$, $N_1 \xrightarrow{*}_\beta N_1'$, and $N_2 \xrightarrow{*}_\beta N_2'$. Since by assumption, $\langle \lambda x\colon \sigma.\ M_1',\ \lambda x\colon \sigma.\ M_2'\rangle \in [\![\sigma \to \tau]\!]$ whenever $M_1 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_1'$ and $M_2 \xrightarrow{+}_\beta \lambda x\colon \sigma.\ M_2'$, and by the induction hypothesis applied at type σ, by (R2) and (R0),[2] $\langle N_1',\ N_2'\rangle \in [\![\sigma]\!]$, we conclude that $\langle(\lambda x\colon \sigma.\ M_1')N_1',\ (\lambda x\colon \sigma.\ M_2')N_2'\rangle \in [\![\tau]\!]$. By the induction hypothesis applied at type τ, by (R2) and (R0), we have $\langle Q_1,\ Q_2\rangle \in [\![\tau]\!]$, and by (R3), we have $\langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]$.

Since $\langle M_1,\ M_2\rangle \in R_{\sigma\to\tau}$ and $\langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]$ whenever $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, we conclude that $\langle M_1,\ M_2\rangle \in [\![\sigma \to \tau]\!]$. ◀

[2] Symmetry and transitivity are needed, but they follow from (R0).

For the proof of the next lemma, we need to add two new conditions (P4) and (P5) to (P0)-(P3).

▶ **Definition 8.** *Properties (P4) and (P5) are defined as follows:*

(P4) *If* $\langle M_1,\ M_2\rangle \in R_\tau$*, then* $\langle \lambda x\colon \sigma.\, M_1,\ \lambda x\colon \sigma.\, M_2\rangle \in R_{\sigma\to\tau}$*.*
(P5) *If* $\langle N_1, N_2\rangle \in R_\sigma$ *and* $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in R_\tau$*, then* $\langle (\lambda x\colon \sigma.\, M_1)N_1, (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_\tau$*.*

▶ **Lemma 9.** *If* $\mathcal{R}$ *is a family of relations satisfying conditions (P0)-(P5) and for every* $\langle N_1, N_2\rangle$*, (*$\langle N_1, N_2\rangle \in [\![\sigma]\!]$ *implies* $\langle M_1[N_1/x],\ M_2[N_2/x]\rangle \in [\![\tau]\!]$*), then* $\langle \lambda x\colon \sigma.M_1, \lambda x\colon \sigma.M_2\rangle \in [\![\sigma \to \tau]\!]$*.*

Proof. We prove that $\langle \lambda x\colon \sigma.\, M_1,\ \lambda x\colon \sigma.\, M_2\rangle \in R_{\sigma\to\tau}$ and that for every every $\langle N_1,\ N_2\rangle$, if $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, then $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in [\![\tau]\!]$. We will need the fact that the sets of the form $[\![\sigma]\!]$ have the properties (R0)-(R3), but this follows from Lemma 7, since (P0)-(P3) hold. First, we prove that $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_{\sigma\to\tau}$.

Since by Lemma 7, $\langle x,\ x\rangle \in [\![\sigma]\!]$ for every variable of type σ, by the assumption of Lemma 9, $\langle M_1[x/x],\ M_2[x/x]\rangle = \langle M_1,\ M_2\rangle \in [\![\tau]\!]$. Then, by (R1), $\langle M_1,\ M_2\rangle \in R_\tau$, and by (P4), we have $\langle \lambda x\colon \sigma.\, M_1,\ \lambda x\colon \sigma.\, M_2\rangle \in R_{\sigma\to\tau}$.

Next, we prove that for every every $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, then $\langle (\lambda x\colon \sigma.M_1)N_1,\ (\lambda x\colon \sigma.M_2)N_2\rangle \in [\![\tau]\!]$. Assume that $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$. Then, by the assumption of Lemma 9, we deduce that $\langle M_1[N_1/x],\ M_2[N_2/x]\rangle \in [\![\tau]\!]$. Thus, by (R1), we have $\langle N_1,\ N_2\rangle \in R_\sigma$ and $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in R_\tau$. By (P5), we have $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_\tau$. Now, there are two cases.

If τ is a base type, then $[\![\tau]\!] = R_\tau$. But we just showed that $\langle (\lambda x\colon \sigma.M_1)N_1, (\lambda x\colon \sigma.M_2)N_2\rangle \in R_\tau$, so we have $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in [\![\tau]\!]$.

If τ is not a base type, then $(\lambda x\colon \sigma.\, M_1)N_1$ and $(\lambda x\colon \sigma.\, M_2)N_2$ are simple. Thus, we prove that $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in [\![\tau]\!]$ using (R3). The case where $(\lambda x\colon \sigma.\, M_1)N_1$ and $(\lambda x\colon \sigma.\, M_2)N_2$ are stubborn is trivial. The case where $(\lambda x\colon \sigma.\, M_1)N_1$ is not stubborn and $(\lambda x\colon \sigma.\, M_2)N_2$ is stubborn is similar to the next case and simpler (and the symmetric case follows by (R0)).

If $(\lambda x\colon \sigma.\, M_1)N_1$ and $(\lambda x\colon \sigma.\, M_2)N_2$ are not stubborn and if $(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{+}_\beta Q_1$ and $(\lambda x\colon \sigma.\, M_2)N_2 \xrightarrow{+}_\beta Q_2$, where $Q_1 = \lambda y\colon \gamma.\, P_1$ and $Q_2 = \lambda y\colon \gamma.\, P_2$ are I-terms, then the reductions are necessarily of the form

$$(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{*}_\beta (\lambda x\colon \sigma.\, M_1')N_1' \longrightarrow_\beta M_1'[N_1'/x] \xrightarrow{*}_\beta Q_1,$$

$$(\lambda x\colon \sigma.\, M_2)N_2 \xrightarrow{*}_\beta (\lambda x\colon \sigma.\, M_2')N_2' \longrightarrow_\beta M_2'[N_2'/x] \xrightarrow{*}_\beta Q_2,$$

where $M_1 \xrightarrow{*}_\beta M_1'$, $M_2 \xrightarrow{*}_\beta M_2'$, $N_1 \xrightarrow{*}_\beta N_1'$, and $N_2 \xrightarrow{*}_\beta N_2'$. But $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in [\![\tau]\!]$, and since

$$M_1[N_1/x] \xrightarrow{*}_\beta M_1'[N_1'/x] \xrightarrow{*}_\beta Q_1,$$

and

$$M_2[N_2/x] \xrightarrow{*}_\beta M_2'[N_2'/x] \xrightarrow{*}_\beta Q_2,$$

by (R2) and (R0), we have $\langle Q_1,\ Q_2\rangle \in [\![\tau]\!]$. Since $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_\tau$ and $\langle Q_1,\ Q_2\rangle \in [\![\tau]\!]$ whenever $(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{+}_\beta Q_1$ and $(\lambda x\colon \sigma.\, M_2)N_2 \xrightarrow{+}_\beta Q_2$, by (R3), we have $\langle (\lambda x\colon \sigma.\, M_1)N_1,\ (\lambda x\colon \sigma.\, M_2)N_2\rangle \in [\![\tau]\!]$. ◀

▶ **Lemma 10.** *Given a family of relations $\mathcal{R}$ satisfying conditions (P0)-(P5), for every pair $\langle M_1,\ M_2\rangle$ of type σ, for every pair of substitutions φ_1 and φ_2 such that $\langle \varphi_1(y),\ \varphi_2(y)\rangle \in [\![\gamma]\!]$ for every $y\colon \gamma \in FV(M_1) \cup FV(M_2)$, if $\vdash_\beta M_1 \doteq M_2\colon \sigma$, then $\langle M_1[\varphi_1],\ M_2[\varphi_2]\rangle \in [\![\sigma]\!]$.*

Proof. First, we prove the lemma, but in the case where $M_1 \doteq M_2\colon \sigma$ is provable in the proof system of Definition 2 **without using** the axioms (β) or (η). We proceed by induction on the proof of $M_1 = M_2$.

$$x \doteq x\colon \sigma \quad (\textit{reflexivity})$$

Obvious, since by assumption, $\langle \varphi_1(x),\ \varphi_2(x)\rangle \in [\![\sigma]\!]$.

$$\frac{M_1 \doteq M_2\colon \sigma}{M_2 \doteq M_1\colon \sigma} \quad (\textit{symmetry})$$

By the induction hypothesis, $\langle M_1[\varphi_1],\ M_2[\varphi_2]\rangle \in [\![\sigma]\!]$. Since by Lemma 7 (R0), every $[\![\gamma]\!]$ is symmetric, we also have $\langle M_2[\varphi_2],\ M_1[\varphi_1]\rangle \in [\![\sigma]\!]$.

$$\frac{M_1 \doteq M_2\colon \sigma \quad M_2 \doteq M_3\colon \sigma}{M_1 \doteq M_3\colon \sigma} \quad (\textit{transitivity})$$

By the induction hypothesis, $\langle M_1[\varphi_1],\ M_2[\varphi_2]\rangle \in [\![\sigma]\!]$ and $\langle M_2[\varphi_2],\ M_3[\varphi_3]\rangle \in [\![\sigma]\!]$. Since by Lemma 7 (R0), every $[\![\gamma]\!]$ is transitive, we also have $\langle M_1[\varphi_1],\ M_3[\varphi_3]\rangle \in [\![\sigma]\!]$.

$$\frac{M_1 \doteq M_2\colon (\sigma \to \tau) \quad N_1 \doteq N_2\colon \sigma}{(M_1N_1) \doteq (M_2N_2)\colon \tau} \quad (\textit{congruence})$$

By the induction hypothesis, $\langle M_1[\varphi_1],\ M_2[\varphi_2]\rangle \in [\![\sigma \to \tau]\!]$ and $\langle N_1[\varphi_1],\ N_2[\varphi_2]\rangle \in [\![\sigma]\!]$. By the definition of $[\![\sigma \to \tau]\!]$, we get $\langle M_1[\varphi_1]N_1[\varphi_1],\ M_2[\varphi_2]N_2[\varphi_2]\rangle \in [\![\tau]\!]$, which shows that

$$\langle (M_1N_1)[\varphi_1],\ (M_2N_2)[\varphi_2]\rangle \in [\![\tau]\!],$$

since $M_1[\varphi_1]N_1[\varphi_1] = (M_1N_1)[\varphi_1]$ and $M_2[\varphi_2]N_2[\varphi_2] = (M_2N_2)[\varphi_2]$.

$$\frac{M_1 \doteq M_2\colon \tau}{\lambda x\colon \sigma.\ M_1 \doteq \lambda x\colon \sigma.\ M_2\colon (\sigma \to \tau)} \quad (\xi)$$

Consider any $\langle N_1,\ N_2\rangle \in [\![\sigma]\!]$, and any substitutions φ_1 and φ_2 such that $\langle \varphi_1(y), \varphi_2(y)\rangle \in [\![\gamma]\!]$ for every $y\colon \gamma \in (FV(M_1) \cup FV(M_2) - \{x\})$. Thus, the substitutions $\varphi_1[x := N_1]$ and $\varphi_2[x := N_2]$ have the property that $\langle \varphi_1(y),\ \varphi_2(y)\rangle \in [\![\gamma]\!]$ for every $y\colon \gamma \in FV(M_1) \cup FV(M_2)$. By suitable α-conversion, we can assume that x does not occur in any $\varphi_1(y)$ or $\varphi_2(y)$ for every $y \in dom(\varphi_1) \cup dom(\varphi_2)$, that N_1 is substitutable for x in M_1, and that N_2 is substitutable for x in M_2. Then, $M_1[\varphi_1[x := N_1]] = M_1[\varphi_1][N_1/x]$ and $M_2[\varphi_2[x := N_2]] = M_2[\varphi_2][N_2/x]$. By the induction hypothesis applied to $\langle M_1,\ M_2\rangle$, $\varphi_1[x := N_1]$, and $\varphi_2[x := N_2]$, we have

$$\langle M_1[\varphi_1[x := N_1]],\ M_2[\varphi_2[x := N_2]]\rangle \in [\![\tau]\!],$$

that is, $\langle M_1[\varphi_1][N_1/x],\ M_2[\varphi_2][N_2/x]\rangle \in [\![\tau]\!]$. Consequently, by Lemma 9,

$$\langle (\lambda x\colon \sigma.\ M_1[\varphi_1]),\ (\lambda x\colon \sigma.\ M_2[\varphi_2])\rangle \in [\![\sigma \to \tau]\!],$$

that is,

$$\langle (\lambda x\colon \sigma.\, M_1)[\varphi_1],\ (\lambda x\colon \sigma.\, M_2)[\varphi_2]\rangle \in [\![\sigma \to \tau]\!],$$

since $(\lambda x\colon \sigma.\, M_1[\varphi_1]) = (\lambda x\colon \sigma.\, M_1)[\varphi_1]$ and $(\lambda x\colon \sigma.\, M_2[\varphi_2]) = (\lambda x\colon \sigma.\, M_2)[\varphi_2]$.

This concludes the proof in the case where $M_1 \doteq M_2 \colon \sigma$ is provable in the proof system of Definition 2 **without using** the axioms (β) or (η). We now show that the lemma holds when the axioms (β) are also used.

We noted (just after Definition 2) that the equation $M \doteq M\colon \sigma$ is provable using the (*reflexivity*) axioms and the rules (*congruence*) and (ξ), for every term M. Thus, by the previous proof, we have that $\langle M[\varphi_1],\ M[\varphi_2]\rangle \in [\![\sigma]\!]$ for every term M of type σ. In particular, this holds for the term $(\lambda x\colon \sigma.\, M)N$, and by (R2), we have

$$\langle ((\lambda x\colon \sigma.\, M)N)[\varphi_1],\ M[N/x][\varphi_2]\rangle \in [\![\tau]\!].$$

But this shows that the lemma also holds for every axiom (β), concluding the proof. ◀

▶ **Theorem 11.** *If $\mathcal{R}$ is a binary relation on λ-terms satisfying conditions (P0)-(P5) listed below*

(P0) *Every relation R_σ is a per, i.e., R_σ is symmetric and transitive;*

(P1) *$\langle x,\ x\rangle \in R_\sigma$, $\langle c,\ c\rangle \in R_\sigma$, for every variable x and constant c of type σ;*

(P2) *If $\langle M_1,\ M_2\rangle \in R_\sigma$ and $M_1 \longrightarrow_\beta M_1'$, then $\langle M_1',\ M_2\rangle \in R_\sigma$;*

(P3) *If M_1 and M_2 are simple, $\langle M_1, M_2\rangle \in R_{\sigma\to\tau}$, $\langle N_1, N_2\rangle \in R_\sigma$, and either $\langle (\lambda x\colon \sigma.\, M_1')N_1,\ M_2N_2\rangle \in R_\tau$ whenever $M_1 \overset{+}{\longrightarrow}_\beta \lambda x\colon \sigma.\, M_1'$ and M_2 is stubborn, or $\langle (\lambda x\colon \sigma.\, M_1')N_1,\ (\lambda x\colon \sigma.\, M_2')N_2\rangle \in R_\tau$ whenever $M_1 \overset{+}{\longrightarrow}_\beta \lambda x\colon \sigma.\, M_1'$ and $M_2 \overset{+}{\longrightarrow}_\beta \lambda x\colon \sigma.\, M_2'$, then $\langle M_1N_1,\ M_2N_2\rangle \in R_\tau$;*

(P4) *If $\langle M_1,\ M_2\rangle \in R_\tau$, then $\langle \lambda x\colon \sigma.\, M_1,\ \lambda x\colon \sigma.\, M_2\rangle \in R_{\sigma\to\tau}$;*

(P5) *If $\langle N_1, N_2\rangle \in R_\sigma$ and $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in R_\tau$, then $\langle (\lambda x\colon \sigma.\, M_1)N_1, (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_\tau$;*

then for every provable equation $\vdash_\beta M_1 \doteq M_2 \colon \sigma$, we have $\langle M_1,\ M_2\rangle \in \mathcal{R}_\sigma$ (in other words, every equation provable in the equational β-theory of $\lambda^{\to}$ satisfies the binary predicate defined by $\mathcal{R}$).

Proof. Apply Lemma 10 to every β-provable equation $M_1 \doteq M_2 \colon \sigma$ and to the pair of identity substitutions, which is legitimate since $\langle x,\ x\rangle \in [\![\gamma]\!]$ for every variable of type γ (by Lemma 7). Thus, $\langle M_1,\ M_2\rangle \in [\![\sigma]\!]$ for every β-provable equation $M_1 \doteq M_2 \colon \sigma$, and thus $\langle M_1,\ M_2\rangle \in \mathcal{R}_\sigma$. ◀

▶ Remark. The proof of Lemma 10 actually shows that each $\mathcal{R}_\sigma$ is reflexive.

As an application of Theorem 19, it is easy to prove strong normalization and the Church-Rosser property for $\longrightarrow_\beta$. To do this consider the relation $\mathcal{R}$ defined as $\langle M_1,\ M_2\rangle \in \mathcal{R}$ iff $M_1 \overset{*}{\longleftrightarrow}_\beta M_2$, and both M_1 and M_2 reduce to the same unique normal form. Properties (P0)-(P5) are easily verified, using the same techniques as in Gallier [6]. Of course, this is a bit of an overkill for the simply-typed λ-calculus.

We now show how to extend the previous results to the $\beta\eta$-equational theory of $\lambda^{\to}$.

3 Adding η-Reduction

The rule of η-reduction is an oriented version of axiom (η):

$$\lambda x\colon \sigma.\,(Mx) \longrightarrow_{\eta} M,$$

where $x \notin FV(M)$. We will denote the reduction relation defined by β-reduction and η-reduction as $\longrightarrow_{\beta\eta}$.

The definition of an I-term remains identical to that given in Definition 3, and similarly for stubborn terms. Properties (P0)-(P3) also remain the same, but they are stated with respect to the new reduction relation $\xrightarrow{+}_{\beta\eta}$.

▶ **Definition 12.** *Properties (P0)-(P3) are defined as follows:*
(P0) *Every relation R_σ is a per, i.e., R_σ is symmetric and transitive.*
(P1) *$\langle x,\ x\rangle \in R_\sigma$, $\langle c,\ c\rangle \in R_\sigma$, for every variable x and constant c of type σ.*
(P2) *If $\langle M_1,\ M_2\rangle \in R_\sigma$ and $M_1 \longrightarrow_{\beta\eta} M_1'$, then $\langle M_1',\ M_2\rangle \in R_\sigma$.*
(P3) *If M_1 and M_2 are simple, $\langle M_1, M_2\rangle \in R_{\sigma\to\tau}$, $\langle N_1, N_2\rangle \in R_\sigma$, and either $\langle(\lambda x\colon \sigma.\ M_1')N_1,\ M_2N_2\rangle \in R_\tau$ whenever $M_1 \xrightarrow{+}_{\beta\eta} \lambda x\colon \sigma.\ M_1'$ and M_2 is stubborn, or $\langle(\lambda x\colon \sigma.\ M_1')N_1,\ (\lambda x\colon \sigma.\ M_2')N_2\rangle \in R_\tau$ whenever $M_1 \xrightarrow{+}_{\beta\eta} \lambda x\colon \sigma.\ M_1'$ and $M_2 \xrightarrow{+}_{\beta\eta} \lambda x\colon \sigma.\ M_2'$, then $\langle M_1N_1, M_2N_2\rangle \in R_\tau$.*

From now on, we only consider families of relations $\mathcal{R}$ satisfying conditions (P0)-(P3) of Definition 12. Definition 5 remains the same, except that it uses the new reduction relation $\longrightarrow_{\beta\eta}$.

▶ **Definition 13.** *For any type σ, a nonempty relation $C \subseteq \Lambda_\sigma \times \Lambda_\sigma$ is a $\mathcal{R}$-logical candidate iff it satisfies the following conditions:*
(R0) *C is a per.*
(R1) *$C \subseteq R_\sigma$.*
(R2) *If $\langle M_1,\ M_2\rangle \in C$ and $M_1 \longrightarrow_{\beta\eta} M_1'$, then $\langle M_1',\ M_2\rangle \in C$.*
(R3) *If M_1 and M_2 are simple, $\langle M_1,\ M_2\rangle \in R_\sigma$, and either $\langle \lambda x\colon \gamma.\ M_1',\ M_2\rangle \in C$ whenever $M_1 \xrightarrow{+}_{\beta\eta} \lambda x\colon \gamma.\ M_1'$ and M_2 is stubborn, or $\langle \lambda x\colon \gamma.\ M_1',\ \lambda x\colon \gamma.\ M_2'\rangle \in C$ whenever $M_1 \xrightarrow{+}_{\beta\eta} \lambda x\colon \gamma.\ M_1'$ and $M_2 \xrightarrow{+}_{\beta\eta} \lambda x\colon \gamma.\ M_2'$, then $\langle M_1,\ M_2\rangle \in C$.*

Definition 6 remains unchanged, but we repeat it for convenience.

▶ **Definition 14.** *The logical relations $[\![\sigma]\!]$ are defined as follows:*

$$\begin{aligned}
[\![\sigma]\!] &= R_\sigma, \qquad \sigma \text{ a base type},\\
[\![\sigma\to\tau]\!] &= \{\langle M_1,\ M_2\rangle \mid \langle M_1,\ M_2\rangle \in R_{\sigma\to\tau},\ \text{and for all } N_1,\ N_2,\\
&\qquad \text{if } \langle N_1,\ N_2\rangle \in [\![\sigma]\!] \text{ then } \langle M_1N_1,\ M_2N_2\rangle \in [\![\tau]\!]\}.
\end{aligned}$$

Lemma 7 also holds.

▶ **Lemma 15.** *If $\mathcal{R}$ is a family of relations satisfying conditions (P0)-(P3), then each $[\![\sigma]\!]$ is a $\mathcal{R}$-logical candidate that contains all pairs of stubborn terms in R_σ.*

Proof. Careful inspection reveals that the proof of Lemma 7 remains unchanged. This is because, for a simple term M:

If $M \in \Lambda_{\sigma\to\tau}$ and there is a reduction $MN \xrightarrow{+}_{\beta\eta} Q$ where Q is an I-term, we must have $M \xrightarrow{+}_{\beta\eta} \lambda x\colon \sigma.\, M_1$, even w.r.t. the reduction relation $\xrightarrow{+}_{\beta\eta}$. ◀

Properties (P4) and (P5) are unchanged, but we repeat them for convenience.

▶ **Definition 16.** *Properties (P4) and (P5) are defined as follows:*

(P4) *If* $\langle M_1,\ M_2\rangle \in R_\tau$*, then* $\langle \lambda x\colon \sigma.\, M_1,\ \lambda x\colon \sigma.\, M_2\rangle \in R_{\sigma\to\tau}$*.*

(P5) *If* $\langle N_1, N_2\rangle \in R_\sigma$ *and* $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in R_\tau$*, then* $\langle(\lambda x\colon \sigma.\, M_1)N_1, (\lambda x\colon \sigma.\, M_2)N_2\rangle \in R_\tau$*.*

Lemma 9 also extends to $\beta\eta$-reduction.

▶ **Lemma 17.** *If* $\mathcal{R}$ *is a family of relations satisfying conditions (P0)-(P5) and for every* $\langle N_1, N_2\rangle$*,* $(\langle N_1, N_2\rangle \in [\![\sigma]\!]$ *implies* $\langle M_1[N_1/x], M_2[N_2/x]\rangle \in [\![\tau]\!])$*, then* $\langle \lambda x\colon \sigma. M_1, \lambda x\colon \sigma. M_2\rangle \in [\![\sigma \to \tau]\!]$*.*

Proof. This time, a few changes to the proof of Lemma 9 have to be made to take η-reduction rules into account.

We need to reexamine the case where

$$(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{+}_{\beta\eta} Q_1$$

and Q_1 is an I-term (and similarly for $(\lambda x\colon \sigma. M_2)N_2 \xrightarrow{+}_{\beta\eta} Q_2)$. The reduction is necessarily of the form either

$$(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{*}_{\beta\eta} (\lambda x\colon \sigma.\, M_1')N_1' \longrightarrow_{\beta\eta} M_1'[N_1'/x] \xrightarrow{*}_{\beta\eta} Q_1,$$

where $M_1 \xrightarrow{*}_{\beta\eta} M_1'$ and $N_1 \xrightarrow{*}_{\beta\eta} N_1'$, or

$$(\lambda x\colon \sigma.\, M_1)N_1 \xrightarrow{*}_{\beta\eta} (\lambda x\colon \sigma.\, (M_1'x))N_1' \longrightarrow_{\beta\eta} M_1'N_1' \xrightarrow{*}_{\beta\eta} Q_1,$$

where $M_1 \xrightarrow{*}_{\beta\eta} M_1'x$, with $x \notin FV(M_1')$, and $N_1 \xrightarrow{*}_{\beta\eta} N_1'$.

The first case is as in Lemma 9, we have

$$M_1[N_1/x] \xrightarrow{*}_{\beta\eta} M_1'[N_1'/x] \xrightarrow{*}_{\beta\eta} Q_1.$$

In the second case, as $x \notin FV(M_1')$, we have $M_1'N_1' = (M_1'x)[N_1'/x]$. Since $M_1 \xrightarrow{*}_{\beta\eta} M_1'x$ and $N_1 \xrightarrow{*}_{\beta\eta} N_1'$, we have

$$M_1[N_1/x] \xrightarrow{*}_{\beta\eta} (M_1'x)[N_1'/x] = M_1'N_1' \xrightarrow{*}_{\beta\eta} Q_1.$$

Thus, in all cases,

$$M_1[N_1/x] \xrightarrow{*}_{\beta\eta} Q_1 \quad \text{and} \quad M_2[N_2/x] \xrightarrow{*}_{\beta\eta} Q_2,$$

and since $\langle M_1[N_1/x],\ M_2[N_2/x]\rangle \in [\![\tau]\!]$, by (R2) and (R0), we have $\langle Q_1,\ Q_2\rangle \in [\![\tau]\!]$. ◀

Since Lemma 15 and Lemma 17 hold, so does the extension of Lemma 10 to $\beta\eta$-provability.

▶ **Lemma 18.** *Given a family of relations* $\mathcal{R}$ *satisfying conditions (P0)-(P5), for every pair* $\langle M_1,\ M_2\rangle$ *of type* σ*, for every pair of substitutions* φ_1 *and* φ_2 *such that* $\langle \varphi_1(y),\ \varphi_2(y)\rangle \in [\![\gamma]\!]$ *for every* $y\colon \gamma \in FV(M_1) \cup FV(M_2)$*, if* $\vdash_{\beta\eta} M_1 \doteq M_2\colon \sigma$*, then* $\langle M_1[\varphi_1],\ M_2[\varphi_2]\rangle \in [\![\sigma]\!]$*.*

Proof. The proof is similar to that of Lemma 10, but we also need to treat the case of the (η)-axioms. Recall that the proof shows that $\langle M[\varphi_1],\ M[\varphi_2]\rangle \in [\![\sigma]\!]$ for every term M of type σ. In particular, this holds for the term $\lambda x\colon \sigma.\,(Mx)$ where $x \notin FV(M)$. By (R2), we have

$$\langle((\lambda x\colon \sigma.\,(Mx))[\varphi_1],\ M[\varphi_2]\rangle \in [\![\tau]\!].$$

This concludes the proof. ◀

▶ **Theorem 19.** *If $\mathcal{R}$ is a binary relation on λ-terms satisfying conditions (P0)-(P5), then for every provable equation $\vdash_{\beta\eta} M_1 \doteq M_2\colon \sigma$, we have $\langle M_1,\ M_2\rangle \in \mathcal{R}_\sigma$ (in other words, every equation provable in the equational $\beta\eta$-theory of $\lambda^{\rightarrow}$ satisfies the binary predicate defined by $\mathcal{R}$).*

Proof. Apply Lemma 18 to every $\beta\eta$-provable equation $M_1 \doteq M_2\colon \sigma$ and to the pair of identity substitutions, which is legitimate since $\langle x,\ x\rangle \in [\![\gamma]\!]$ for every variable of type γ (by Lemma 15). Thus, $\langle M_1,\ M_2\rangle \in [\![\sigma]\!]$ for every $\beta\eta$-provable equation $M_1 \doteq M_2\colon \sigma$, and thus $\langle M_1,\ M_2\rangle \in \mathcal{R}_\sigma$, ◀

Several variations of Lemma 18 and Theorem 19 are possible. We can use $\beta\eta$-convertibility instead of $\beta\eta$-reduction in Definition 12, Definition 13, and Definition 16. We can drop symmetry from (R0) and (P0), or drop (R0) and (P0) altogether. In these last two cases, we obtain a version of Lemma 18 by suitably restricting provability. Further investigations are required.

As in the case of β-conversion, it is possible to prove strong normalization and the Church-Rosser property for $\longrightarrow_{\beta\eta}$, using Theorem 19. To do this consider the relation $\mathcal{R}$ defined as $\langle M_1,\ M_2\rangle \in \mathcal{R}$ iff $M_1 \overset{*}{\longleftrightarrow}_{\beta\eta} M_2$, and both M_1 and M_2 reduce to the same unique normal form. Properties (P0)-(P5) are easily verified.

Obviously, it would be interesting to find more general conditions than properties (P0)-(P5) for which our theorems still hold. We leave this an an open problem.

References

1 S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51:1–77, 1991.
2 V. Breazu-Tannen and T. Coquand. Extensional models for polymorphism. *Theoretical Computer Science*, 59:85–114, 1988.
3 P. Freyd, P. Mulry, G. Rosolini, and D. Scott. Extensional pers. *Information and Computation*, 98(2):211–227, 1992.
4 H. Friedman. Equality between functionals. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Math.*, pages 22–37. Springer-Verlag, 1975.
5 Jean H. Gallier. On Girard's "candidats de reductibilité". In P. Odifreddi, editor, *Logic And Computer Science*, pages 123–203. Academic Press, London, New York, May 1990.
6 Jean H. Gallier. On the correspondence between proofs and λ-terms. In P. DeGroote, editor, *The Curry-Howard Isomorphism*, Cahiers du Centre de Logique, No. 8, pages 55–138. Université Catholique de Louvain, 1995.
7 Jean H. Gallier. Proving properties of typed λ-terms using realizability, covers, and sheaves. *Theoretical Computer Science*, 142:299–368, 1995.
8 Jean H. Gallier. Typing untyped lambda terms, or reducibility strikes again! *Annals of Pure and Applied Logic*, 91:231–270, 1998.
9 Jean-Yves Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J.E. Fenstad, editor, *Proc. 2nd Scand. Log. Symp.*, pages 63–92. North-Holland, 1971.

10 Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Université de Paris VII, June 1972. Thèse de Doctorat d'Etat.

11 Gérard Huet. Initiation au λ-calcul. Technical report, Université Paris VII, Paris, 1991. Lectures Notes.

12 J.M.E. Hyland. The effective topos. In A. S. Troelstra and D. Van Dalen, editors, *L. E. J. Brouwer, Centenary Symposium*, Studies in Logic. North-Holland, 1982.

13 G. Koletsos. Church-Rosser theorem for typed functional systems. *J. Symbolic Logic*, 50(3):782–790, 1985.

14 J.L. Krivine. *Lambda-Calcul, types et modèles.* Etudes et recherches en informatique. Masson, 1990.

15 J. C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In *ACM Conference on LISP and Functional Programming*, pages 308–319. ACM, 1986. Reprinted in *Logical Foundations of Functional Programming*, G. Huet, Ed., Addison Wesley, 1990, 195-212.

16 J.C. Mitchell and E Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51:99–124, 1991.

17 P. S. Mulry. Generalized Banach-Mazur functionals in the topos of recursive sets. *J. Pure Appl. Alebra*, 26, 1982.

18 G.D. Plotkin. Lambda definability in the full type hierarchy. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–373, London, 1980. Academic Press.

19 R. Statman. Completeness, invariance, and λ-definability. *J. Symbolic Logic*, 47(1):17–26, 1982.

20 R. Statman. Equality between functionals, revisited. In Harrington, Morley, Scedrov, and Simpson, editors, *Harvey Friedman's Research on the Foundations of Mathematics*, pages 331–338. North-Holland, 1985.

21 R. Statman. Logical Relations and the Typed λ-Calculus. *Information and Control*, 65(2/3):85–97, 1985.

22 W.W. Tait. Intensional interpretation of functionals of finite type I. *J. Symbolic Logic*, 32:198–212, 1967.

23 W.W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium*, volume 453 of *Lecture Notes in Math.*, pages 240–251. Springer Verlag, 1975.

AutoML for Explainable Anomaly Detection (XAD)

Nikolaos Myrtakis
Department of Computer Science, University of Crete, Heraklion, Greece
ETIS Laboratory, CY Cergy Paris Université, ENSEA, France

Ioannis Tsamardinos
Department of Computer Science, University of Crete, Heraklion, Greece

Vassilis Christophides
ETIS Laboratory, CY Cergy Paris Université, ENSEA, France

Abstract

Numerous algorithms have been proposed for detecting anomalies (outliers, novelties) in an unsupervised manner. Unfortunately, it is not trivial, in general, to understand why a given sample (record) is labelled as an anomaly and thus diagnose its root causes. We propose the following *reduced-dimensionality, surrogate model approach* to explain detector decisions: approximate the detection model with another one that employs only a small subset of features. Subsequently, samples can be visualized in this low-dimensionality space for human understanding. To this end, we develop **PROTEUS**, an AutoML pipeline to produce the surrogate model, specifically designed for feature selection on imbalanced datasets. The PROTEUS surrogate model can not only explain the training data, but also the out-of-sample (unseen) data. In other words, PROTEUS produces **predictive** explanations by approximating the decision surface of an unsupervised detector. PROTEUS is designed to return an accurate estimate of out-of-sample predictive performance to serve as a metric of the quality of the approximation. Computational experiments confirm the efficacy of PROTEUS to produce predictive explanations for different families of detectors and to reliably estimate their predictive performance in unseen data. Unlike several ad-hoc feature importance methods, PROTEUS is robust to high-dimensional data.

2012 ACM Subject Classification Computing methodologies → Anomaly detection

Keywords and phrases Anomaly Explanation, Predictive Explanation, Anomaly Interpretation, Explainable AI

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.8

Funding The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant” (Project Number: 1941); the ERC under the European Union’s Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 617393.

1 Introduction

Detection of “anomalous” samples (records, instances), called *anomaly detection*, is an important problem in machine learning. It is conceptually related to outlier and novelty detection in several application settings. The anomalous samples may indicate mislabelled data, catastrophic measurements or data entry errors, bugs in data wrangling and preprocessing software, or other interesting phenomena.

Numerous **unsupervised** algorithms (e.g., IF [32], LOF [7], LODA [44]) to detect anomalies (hereafter **detectors**) have been proposed. The most advanced ones detect anomalies in a multi-dimensional fashion, simultaneously considering all feature values. Unfortunately, detectors, in general, do not explain why a sample was considered as abnormal, leaving human analysts with no guidance about their root causes, insight to take corrective actions, or remedy their effect.

Several methods for **explaining anomalies** have been proposed, hereafter **explainers**. *The explanations often take the form of a subset of features* called a **subspace** in the literature. The idea is that *by examining only the explaining features suffices to determine whether the sample is an anomaly or not according to the detector.*

Existing methods can be categorized to those that provide **local explanations** (point-based) that pertain to a single sample, or **global explanations** (a.k.a. set-based) to simultaneously explain all training samples. The latter is important in order to reduce the burden of human analysts to inspect possibly different explanations for each anomaly. We should stress that global explanation is different from clustering as the former's objective is to provide a subspace segregating the anomalous from normal samples. Explainers may be **specific** to a detection algorithm or detector-**agnostic**, hence applicable post-hoc to any detection algorithm. As reported by several independent experimental studies, e.g. [17], there is no detector outperforming all others on all possible datasets. Hence, researchers cannot just design a specific explainer for the optimal detector; it may thus be preferable to design optimal agnostic explainers. Explainers may also be categorized as **descriptive** in the sense that they explain the samples used to train the detector. Explainers that return explanations that generalize to unseen data are **predictive** ones. The importance of predictive explanations has been recognised in Explainable AI to avoid recomputing explanations on every new batch of data.

Figure 1 illustrates how predictive explanations can be used in data validation pipelines monitoring the data fed to downstream ML models. Given that in real application settings it is difficult or even impossible to label data as anomalous or normal [17], unsupervised detectors are initially used to spot anomalies. Then, a predictive anomaly explainer could be used by human analysts to reveal the root causes of the detected anomalies and decide subsequent corrective actions. It is essentially a surrogate model [1], trained with a small subset of the original features that serve as explaining feature subspace. Depending on the quality of the approximation of the decision boundary of an unsupervised detector, the surrogate model can be also used to detect anomalies in fresh data, i.e., new batches of data, by completely bypassing the need to rerun the detector.

In this paper, we propose **a novel method to produce global, predictive explanations** called **PROTEUS**[2]. PROTEUS is *detector agnostic*, and can be used to approximate the decision boundary of any detector. We should stress that prior work on detector agnostic explainers like CA-Lasso [38] and SHAP [34] but also detector specific explainers like LODA [44] produce explanations that are only **local and descriptive**.

PROTEUS essentially constructs a *reduced-dimensionality, surrogate model* that approximates the behavior of a detector with fewer features. Since the detector is labelling the samples as anomalies or not, the problem of finding such a model reduces to a *supervised predictive modeling with feature selection* problem. In order for the surrogate model to also explain unseen samples, it has to approximate the detector's decision boundary and not simply interpolate the anomalies (overfit) in the training data. To this respect, *the quality of approximation should be estimated using out-of-sample performance estimation protocols* like K-fold cross validation (CV). To build the model, any combination of feature selection algorithm with a classifier could be employed. However, ideally one should optimize the combination of algorithms and their hyper-parameter values to achieve the best approximation with the samples at hand.

[1] A surrogate model is an interpretable model that is trained to approximate the predictions of a black box model [39]

[2] Proteus or *Πρωτευs* in Greek, means "first" and is a minor sea God and son of Poseidon.

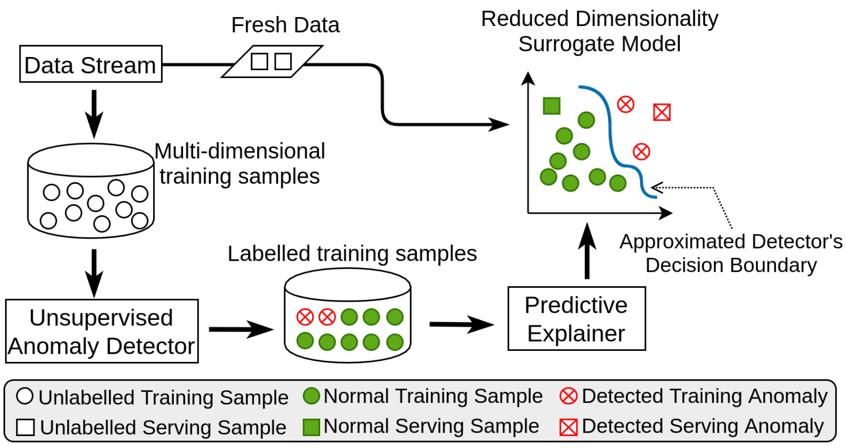

Fresh Data
Reduced Dimensionality Surrogate Model
Data Stream
Multi-dimensional training samples
Approximated Detector's Decision Boundary
Labelled training samples
Unsupervised Anomaly Detector
Predictive Explainer
Unlabelled Training Sample
Normal Training Sample
Detected Training Anomaly
Unlabelled Serving Sample
Normal Serving Sample
Detected Serving Anomaly

Figure 1 Predictive Anomaly Explanation Pipeline.

The above requirements for tuning and estimating generalization performance of predictive models are nowadays addressed in the automated machine learning (**AutoML**) systems [23]. In this respect, **producing predictive anomaly explanations can be solved as an AutoML problem**. Unfortunately, the majority of existing tools such as auto-sklearn do not perform feature selection. In addition, they do not exploit the fact that the data can be augmented with new samples (pseudo-samples) that can be labelled by the detector, to improve performance. Finally, their performance estimates are often overestimated [51], particularly for imbalanced datasets. To address the above issues, PROTEUS makes the following contributions:

(1) In Section 2, we introduce a novel AutoML engine specifically designed to support feature selection and classification on imbalanced datasets. Unlike existing explainers, PROTEUS outputs not only a *small-sized feature subset serving as explanation* but also a *surrogate model fitted on this subset* to explain unseen samples, as well as a *reliable out-of-sample (predictive) performance estimation.*

(2) To produce such output, PROTEUS AutoML relies on *advanced design choices* described in Section 3, such as supervised oversampling, group-based stratification, and a special variant of Cross-Validation with Bootstrap Bias Correction (**BBC**) [53].

(3) Thorough computational experiments presented in Section 4 we show the efficacy and robustness of PROTEUS in synthetic and real datasets of increasing dimensionality. Last but not least, our experiments show that PROTEUS approximates accurately the performance of a specific explainer (LODA) in a detector-agnostic fashion.

This is a substantial extension of our short paper published at ICDE 2021 [41]. This paper's additional contributions are four-fold:

(4) We formally define in Section 2 descriptive and predictive explanations originally introduced in our work.

(5) We assess in Section 5 the merit of the idea to use PROTEUS to correct the decisions of the unsupervised anomaly detectors. Specifically, we study the disagreements of classification to anomalies between the surrogate PROTEUS model and the detector. We show that PROTEUS can often correct the false positives of false negatives as identified by the detector.

(6) We propose a new visualization method for presenting the global explanations found by PROTEUS as spider charts. The visualizations provide insight regarding the combination of feature values that lead to calling a sample as anomalous or not.

(7) We survey in Section 6 several categories of related work on explaining anomalies in unsupervised and supervised settings, positioning the predictive explanation of PROTEUS w.r.t. each category.

Finally, in Section 7 we conclude the paper and discuss directions of future research.

2 Problem Definition

In this section, we formalize the notion of descriptive explanations inspired by [19] and we introduce the novel concept of predictive explanations.

Let $D = \{x_1, \ldots, x_n\}$ be a dataset of n samples, where each sample $x \in \mathbb{R}^d$. An **Anomaly Detector** A is essentially a function that scores the "anomalousness" of samples in D according to an unsupervised **Anomaly Model**: $\omega_A : \mathbb{R}^d \to \mathbb{R}$. Continuous scores are then converted into dichotomous decisions using a threshold choice method [55]. Given a threshold T and sample $x \in D$, a **binary** Anomaly detector is a function $\omega_A^l : \mathbb{R} \to \{0, 1\}$ defined as follows: $\omega_A^l(x) = \mathbb{1}[\omega_A(x) > T]$. The value $\omega_A^l(x) = 1$, semantically denotes the identification of an anomaly.

▶ **Definition 1.** *The descriptive explanation $\mathscr{D}$ of a set of anomalies $O = \{x \mid \omega_A^l(x) = 1, x \in D\}$, is a subset of features S, where $|S| = b \ll d$, that maximizes the* **cumulative score** *for a set of anomalies:*

$$\mathscr{D} = \underset{S}{\operatorname{argmax}} \sum_{x \in O} \omega_A(x[S]) \qquad \text{s.t.} \quad |S| = b \tag{1}$$

where $[\cdot]$ denotes the projection of x over the features in S composing its explanation.

A global descriptive explanation algorithm strives to reveal the subspace that maximizes the cumulative anomaly score for a set of identified anomalies, given a specific anomalousness criterion such as distance, isolation etc. Such explanations are called *descriptive* as they are computed for every new batch of anomalous and normal samples. In order to make explanations also *discriminative* for unseen data, we need to consider *predictive* explanations i.e., a hyperplane of reduced dimensionality that separates the anomalies from the normal samples when training a classifier over the output of an unsupervised anomaly detector. To produce explaining hyperplanes we need to evaluate alternative surrogate models built using different classification algorithms $h \in H$, where h is fitted in a lower dimensional space, produced in turn by different feature selection algorithms $g \in G$ that consider the labels returned by different anomaly detectors.

In a nutshell, *predictive* explanations are produced by solving an AutoML problem [23]. We denote the combination of an algorithm g and h with their respective hyper-parameter values a and b as a configuration θ, which is a function $f = h(g(\cdot, a), b)$. The function f first applies the specified feature selection algorithm g with hyper-parameters a to some input data and the result is then used to train a classifier h with hyper-parameters b. Let $D^l = \{(x_1, \hat{y}_1), ..., (x_n, \hat{y}_n)\}$ be the augmented dataset D enriched with the anomaly labels as indicated by the detector model: $\hat{y}_i = \omega_A^l(x_i)$.

▶ **Definition 2.** *The predictive explanation $\mathscr{P}$ is the hyperplane that comprises of a minimal subset of features S leading to an optimal surrogate model h w.r.t. a performance metric Q:*

$$\mathscr{P} = \underset{S}{\operatorname{argmax}} \max_{h} Q(h(D^l[S]))$$

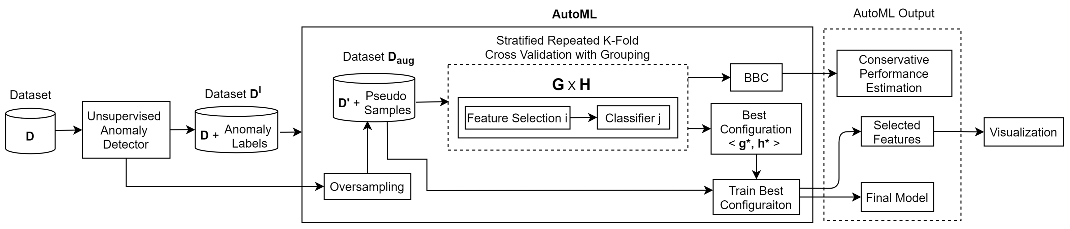


Figure 2 Proteus AutoML Pipeline for Anomaly Detection and Explanation.

Given the dataset D^l, the objective is to build a reduced-dimensionality surrogate model f trained with some data D^l_{train} to *best approximate the detector's decision boundary*. To assess the quality of the approximation, f has to generalize to unseen data D^l_{test} which were not used during the training of f. Therefore, the objective is to find the configuration θ^* that contains the tuple $\langle h^*, g^*, a^*, b^* \rangle$ maximizing a performance metric Q:

$$\theta^* = \underset{\theta}{\text{argmax}} \ Q(f(D^l_{train}, \theta), \ D^l_{test}) \tag{2}$$

The last step is to train the best configuration using all available data, to produce the final surrogate model $f(D^l, \theta^*)$ i.e., a model $h^*(D^l[S], b^*)$ that is used to predict the "anomaloussness" of unseen samples using only a subset of features $S = g^*(D^l, a^*)$.

As anomalies are rare, the quality of performance of a predictive explanation requires evaluation metrics that are insensitive to the class distribution. In this respect, PROTEUS relies on optimizing the area under the Receiver Operating Characteristic (ROC AUC) curve (hereafter **AUC**). Given a minimal subset of features and a classifier, *AUC equals the probability that the classifier scored higher an anomalous than a normal sample.* Discovering such minimal subset is a challenging task as the search space is exponential and features in the input dataset may be both *irrelevant* or *redundant* w.r.t. to the predictive outcome. PROTEUS relies on effective and efficient *feature selection* algorithms [31, 52, 50] to extract predictive explanations in a supervised setting.

3 Producing Global, Predictive Explanations with PROTEUS

Figure 2 illustrates the main steps of the pipelines automatically generated by PROTEUS. We proceed with explaining each step as well as the underlying design choices.

Producing Predictive Explanations as a Supervised Task. First, the anomaly detector runs in dataset D for producing the anomaly scores which are then transformed into binary labels (anomaly or not) in dataset D^l. Producing a surrogate model of lower dimensionality becomes a supervised, binary classification task with feature selection, where the outcome is the label of the unsupervised detector. We note that *data are standardized* for subsequent steps.

Oversampling. D^l is expected to be highly imbalanced (w.r.t. the outcome), as anomalies are rare. Imbalanced datasets are statistically challenging for any ML classifier. One technique to alleviate the problem is *oversampling* the minority class. We focus on *synthetic minority oversampling*, i.e., the samples are perturbed by adding noise to the values of the features, creating new samples called *pseudo-samples*. In common (unsupervised) oversampling methods, for small enough perturbations the pseudo-samples are *assumed* to remain in the minority class. An assumption that strongly depends on the definition of what is considered "small-enough". However, one can take advantage of the detector model produced in the first step is available to query regarding the label of a pseudo-sample. In other words, PROTEUS oversampling is *supervised* as in case of explanation methods for black-box predictive models [45]. Intuitively, oversampling probes the region around the anomalies and perturbs these samples to examine if they cross the detector's decision boundary or not. It thus effectively increases the available sample size for the classification, potentially increasing the quality of the approximation with the surrogate model. For each anomalous sample a it produces ps pseudo-samples per anomaly by adding a perturbation vector p to a: $a' \leftarrow a + p$. Each p follows a multi-variate (d-dimensional) normal distribution with zero mean and an isotropic, diagonal, covariance matrix σI; σ is a hyper-parameter of the algorithm which we set to 0.1 for all the computational experiments. If a' is labelled as an anomaly it is appended to the oversampled dataset D_{aug}, otherwise, another pseudo-sample is produced.

Hyper-Parameter Optimization Space. To produce small-sized explanations, PROTEUS relies on feature selection algorithms, while to produce the surrogate model, a classifier is required. Most classification algorithms also accept a set of hyper-parameter values that also need to be tuned. We will call a combination of feature selection and classification algorithms and their hyper-parameters values as a *configuration. Each configuration is a pipeline that accepts a dataset and produces a classification model and corresponding selected features.* PROTEUS searches the configuration space for the one that leads to an optimal model by performing a simple grid search. This is, the search space of configurations is formed by the Cartesian product $\mathcal{G} \times \mathcal{H}$ (see Figure 2) where $\mathcal{G}$ ($\mathcal{H}$, respectively) is the set of all feature selection (classification) algorithms with bounded hyper-parameter values. As our choices for feature selection algorithms, we include the Statistical Equivalent Signatures (SES) [31], Forward-Backward with Early Dropping (FBED) [52], and Lasso. All of them guarantee to return the optimal feature subset (Markov Blanket in Bayesian Networks) under certain broad (but different for each algorithm) conditions, removing not only *irrelevant*, but also *redundant* features. In general, SES and FBED tend to return smaller feature subsets than Lasso, with a small drop in predictive performance [52].

Moreover, as anomaly explanation targets human analysts, *we limit the number of features selected up to 10*, ranking them based on their score given by the corresponding algorithm (e.g. Lasso coefficients). We selected linear as well as non-linear classifiers considering two facts (a) the extensive experimental results of [12], (b) the fact that deep neural network architectures are almost certain to overfit in very low sample sizes, both in terms of total sample size and the size of the rare class. The present selection of classifiers comprises of: (i) Support Vector Machines, (ii) Random Forest and (iii) K-Nearest-Neighbors. Due to space constraints, we report the hyper-parameters in our GitHub repository[3]. Finally, the number of pseudo-samples to create per anomaly, called ps is also tuned as a hyper-parameter taking values in $\{0, 3, 10\}$. Of course, additional classifiers and feature selection algorithms can be easily integrated in PROTEUS. In total, PROTEUS tried 1800 configurations.

Estimating Performance for Tuning. What is considered as the optimal configuration, out of all tried, is *the one that leads to models with the highest expected out-of-sample (unseen samples) predictive performance.* It is important to estimate this quantity accurately, i.e., with small variance. *A smaller variance of estimation increases the probability that the*

[3] `https://github.com/myrtakis/PROTEUS`

truly optimal configuration will be selected, and thus improves the quality of the final model. Estimation is challenging when there are only few anomalies in the dataset. Indicatively, the synthetic dataset used in our experiments (see Section 4.1) contains 10 anomalies out of 867 samples.

To estimate the expected out-of-sample performance, PROTEUS employs a *Stratified, R-Repeated K-fold Cross Validation with Grouping* protocol. We now explain each part of the protocol. We assume that the reader is familiar with the Standard K-fold Cross Validation (**CV**, hereafter). The *Stratified CV* is a variant where the partitioning to folds is performed under the constraint that the distribution of the classes in each fold is approximately the same as the one in the full dataset [53]. Stratification reduces the variance of estimation for imbalanced data and classes with very few samples (*ibid*). To further reduce the variance of estimation we repeat the CV process multiple times R and take the average (*R-Repeated CV*). Multiple repeats reduce the variance component due to the stochasticity of the specific partitioning [53]. Finally, we come to *Grouping*. By CV with Grouping we indicate a variant of CV that handles grouped samples (a.k.a. as clustered samples in statistics, not to be confused with clustering of samples). These are samples that are not independently sampled and may be correlated given the data distribution. Such samples are repeated measurements on the same subject, as an example. In our context, *an anomaly and its pseudo-samples are grouped*: information from a pseudo-sample in the training set *leaks* to predicting the corresponding anomaly in the test fold. To avoid information leakage, CV with grouping partitions to folds with the constraint that all samples of a group remain in the same fold. In our experiments, we set the number of folds $K = 10$ and the repeats $R = 5$. Hence, each application of the current version of PROTEUS trains $(K \cdot R \cdot \#\ \text{Configurations} + 1) \cdot \mathit{ps} = 90,003$ models.

Producing the Final Surrogate Model and Feature Subset. The final model is trained using all available samples (the full D_{aug}) with the best configuration found, denoted with $\langle F^*, C^* \rangle$ in Figure 2. This configuration also produces the final subset selection (anomaly explanation). The reasoning is that most algorithms (and hence, configurations) are expected to produce better quality models and improved feature selection with more available sample. The models trained during the CV are only employed for selecting the optimal configuration and providing estimates.

Estimating the Out-of-Sample Performance. We now consider how the performance estimate of the final model is produced. Let us assume that 1000 configurations are tried and the best found has a CV estimate of 0.90 AUC. Unfortunately, *the CV estimate of the best configuration is optimistic and should not be returned*, i.e., the actual AUC is expected to be lower. The reason is that our estimate is the best out of 1000 tries [52, 24]. The phenomenon is conceptually similar to the multiple hypothesis testing problem in statistics. In small sample sizes, the over-optimism is particularly striking. Recent work shows that most AutoML tools do not correct for this optimism [51]. In this respect, we apply the Bootstrap Bias Correction (**BBC**, hereafter) to our CV estimates [53] that corrects for this optimism. *This leads to returning conservative estimates of performance on average.*

4 Experimental Evaluation

PROTEUS was implemented in Python 3.6 and evaluated on several synthetic and real-world datasets described subsequently. The code and the datasets used in our experiments are available in our GitHub repository. All experiments were performed in a Linux Desktop computer with a 4-core Intel i5 processor and 32GB of memory.

Table 1 Characteristics of datasets and AUC performance of detectors during training. We denote the parent synthetic dataset as P. Synthetic, the number of features and samples as #F and #S and the anomaly ratio as A.R.

Dat. Name	**#F**	**#S**	**A.R.**	**IF**	**LOF**	**LODA**
P. Synthetic	5	867	1%	0.96	1.0	0.92
W. Br. Cancer	30	377	5%	0.95	0.94	0.96
Ionosphere	33	358	36%	0.85	0.93	0.87
Arrhythmia	257	452	15%	0.80	0.74	0.75

4.1 Synthetic and Real Datasets

We focus on datasets where the samples are *independent and identically distributed (i.i.d.)* and contain numerical features. We employ a *synthetic* dataset, where anomalies have been simulated so that a minimal, global, predictive explanation (feature subset) is both achievable and known. The presence of this gold-standard allows us to evaluate how well PROTEUS identifies it. Specifically, we selected randomly one of the 100-dimensional datasets introduced in [25]. Some anomalies have been generated in a way that makes them outliers according to a subset of 2 of these features, call it S_{2d}, and some according to a subset with 3 (other) features, call it S_{3d}. Thus, the subset of these 5 features $S = S_{2d} \cup S_{3d}$ forms the *gold-standard of global explanation for all anomalies*. On this *parent* synthetic dataset, we added irrelevant features with randomly selected values following a normal distribution with zero mean and standard deviation of one. We ended up with 5 synthetic datasets having 20, 40, 60, 80 and 100 dimensions. All of them contain 867 samples with 10 anomalies i.e., the anomaly ratio is $\approx 1\%$. Such datasets have been frequently used in the literature of anomaly explanation [38, 10, 26, 43], because: (a) the features in an explaining subspace (e.g, S_{2d}) are correlated so they cannot be selected independently; (b) anomalies are recognized as such either in S_{2d} or S_{3d}, but in no other strict subset. Thus, only multivariate detection algorithms and corresponding models will achieve high performance. Hence, PROTEUS must approximate a potentially more complex model.

We additionally consider *real-world datasets* that are widely-used in the evaluation of anomaly detectors. Specifically, we selected the Wisconsin-Breast Cancer, Ionosphere and Arrhythmia, originally from the UCI Machine Learning repository, as defined for anomaly detection purposes in Outlier Detection DataSets (ODDS) repository[4]. They were chosen to ensure that the detectors employed achieve reasonable performance, and thus explanation makes sense. The dataset characteristics and detector performances are shown in Table 1. Wisconsin-Breast Cancer and Ionosphere contain two classes. The minority classes in both datasets are considered as anomalies. For Arrhythmia, eight sub-classes were merged to form the anomaly class. Finally, we added irrelevant features following the procedure described in synthetic datasets constructing three additional datasets per real-world dataset with 30%, 60% and 90% irrelevant feature ratio.

4.2 Experimental Setting

In our experiments, we selected three widely-used unsupervised anomaly detectors that employ different anomalousness criteria, namely Local Outlier Factor (LOF) [7] as a representative of *density-based*, Isolation Forest (IF) [32] as a representative of *isolation-based* and Lightweight On-line Detector of Anomalies (LODA) [44] as a representative of *projection-based* detectors.

[4] `http://odds.cs.stonybrook.edu/`

Regarding the hyper-parameters, for IF we used 100 trees and 256 sub-sample size, for LOF we used $K = 15$ and for LODA we used 100 projection vectors as proposed by the respective authors. To assess the predictive power of a surrogate model produced by PROTEUS we stratified and splitted each dataset into 70% for training and 30% was held out for testing. In each dataset, the detectors run on training and test set before adding irrelevant features. The anomaly threshold T is set as the anomaly ratio for each dataset. The detectors performances are demonstrated in Table 1.

4.3 Feature Importance Alternatives

We compare the original PROTEUS system, employing feature selection methods (call it PROTEUS$_{fs}$), with the PROTEUS pipeline instantiated only with feature importance methods from related explanation methods. We note that these alternatives have been developed to provide *descriptive* explanations; within the PROTEUS pipeline, they are coupled with a classification model, hyper-parameter values are optimized, and they are turned into predictive explanations.

The research question to study is whether *methods specifically developed for explanations in the form of feature importance scores offer additional advantages over the feature selection methods*, everything else being equal (i.e,. the rest of the PROTEUS pipeline). All alternative methods produce *local* explanations, i.e., for individual samples. Importance scores for a given feature are calculated for each sample (local scores). We compute the local scores only for the anomalous samples. To incorporate them into PROTEUS and select features for global explanations, the local scores are averaged out for each feature to produce a final feature importance score, as proposed in [33]. As a final feature selection, we select the top-K features with the highest importance scores. In our experiments, K is set to 10, which is the maximum number of features allowed to be selected by PROTEUS$_{fs}$ and the feature importance methods. Regarding the hyper-parameters for the feature importance alternatives, we used the ones proposed by the respective authors. We evaluate the following alternatives:

(1) Lightweight On-line Detector of Anomalies or **LODA**, hereafter, [44] is an anomaly detector that also returns local feature importance scores. LODA is included as it has shown an excellent trade-off between computational efficiency and anomaly detection performance as a detector [37]. As a feature importance method is selected as a **representative of a detector-specific explanation method**. As such, the results of its explanation method are shown only for the experiments where LODA is also used as the detector. We should stress that when comparing with LODA, the objective is to approximate its performance as the explanation is strongly coupled to the detection process. The resulting PROTEUS variant is called PROTEUS$_{LODA}$.

(2) Kernel **SHAP** (stands for SHapley Additive exPlanations) [34] is a model-agnostic method for local explanation of predictive models producing local feature scores. It is considered state-of-the-art, having outperformed LIME [45]. As Kernel SHAP does not produce a predictive model itself we consider it as a descriptive method. We use the proposed kernel as in the original publication of SHAP. Kernel SHAP is included as a representative of a **model-agnostic feature importance** method, leading to the variant PROTEUS$_{SHAP}$.

(3) **CA-Lasso** [38], is a representative of a **model-agnostic, local feature importance specifically pertaining to anomaly explanation**. It selects k-nearest neighbors per outlier a_i and k other random samples. To overcome the class imbalance, the authors oversample a_i adding pseudo-samples around it, labelling them as anomalies by assumption, until the two classes are balanced. The explanation problem is then turned into binary classification per outlier solved with Lasso. The feature importance of each feature for a_i corresponds to the Lasso coefficients. Rather than learning the decision boundary of individual anomalies PROTEUS builds a binary classifier to explain all the anomalies spotted by an unsupervised detector. In that sense, feature selection in [38] generates local explanations per anomaly that do not generalize to unseen anomalies. Moreover, PROTEUS oversampling is supervised (by the detector) while numerous feature selection and classification algorithms along with the optimization of their hyper-parameter values. Finally, out-of-sample (predictive) performance is estimated by PROTEUS using AUC for subset selection instead of accuracy as originally proposed in [38]. The resulting PROTEUS variant is called PROTEUS$_{CA-Lasso}$.

4.4 PROTEUS Performance Estimation

The objective of this experiment is to assess the effect of PROTEUS design choices, specifically the BBC and Grouping, to provide an accurate performance estimation. Figure 3 depicts the train estimates and test performance when PROTEUS is employed with the design choices described in Section 3, i.e., BBC and CV with Grouping. The dashed black diagonal line indicates the zero bias: points above the diagonal indicate underestimation (negative bias) and below overestimation (optimistic bias). To show the accuracy of the estimation of PROTEUS design choices, we fit a loess curve[5] on train and test performances for every combination (258 in total) of datasets (synthetic and real), detectors (IF, LOF and LODA) and feature selection methods (general purpose and feature importance methods). Ideally, we would want the loess curve to fit exactly the diagonal. Observe that with lower AUC performances PROTEUS tends to overestimate while with higher performances PROTEUS returns a more conservative estimation. In both cases, the points are close to the ideal diagonal line.

To further show the efficacy of the proposed design choices to provide an accurate performance estimation, in Figure 4 we compare the loess curves for train and test estimates for (i) BBC and Grouping (our design choices), (ii) no BBC (i.e., CV estimate) and Grouping (iii) BBC and no Grouping and (iv) no BBC and no Grouping. To quantify the bias for each of the four alternatives, we use the Residual Sum of Squares (RSS) to measure the discrepancy between the train and test performance. When PROTEUS is employed with BBC and Grouping (i), it gives the most accurate estimation of out-of-sample performance (with $RSS_{(i)} = 0.05$) than when using any of the three alternative design choices (with $RSS_{(ii)} = 0.88$, $RSS_{(iii)} = 0.11$ and $RSS_{(iv)} = 0.25$).

4.5 Relevant Features Identification Accuracy

The goal of this experiment is to verify whether the features discovered during the training phase by PROTEUS$_{fs}$ and the feature importance alternatives are part of the gold-standard feature subset S. For this experiment we used the *synthetic* datasets. To assess the quality of the global explanation E in terms of features, we compute $precision(S, E) = \frac{|S \cap E|}{|E|}$ and $recall(S, E) = \frac{|S \cap E|}{|S|}$. As we select the top-10 features to form the explanation and S contains 5 features, the *precision* for the feature importance alternative methods will be up to 0.5. The recall and precision curves are depicted in Figure 5. Feature selection

[5] https://en.wikipedia.org/wiki/Local_regression

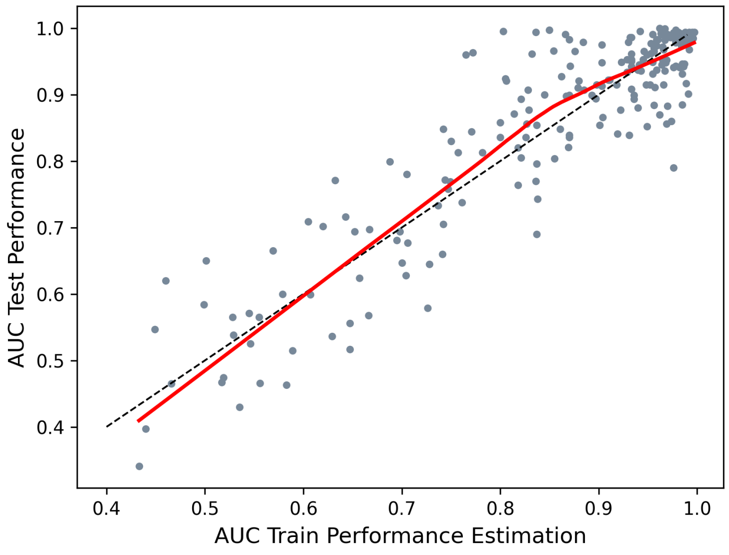


Figure 3 Bias between train and test AUC performances of PROTEUS implemented with BBC and CV with grouping.

methods employed by PROTEUS$_{fs}$ exhibit the highest precision never dropping below 0.5, independently of the employed detector or dataset dimensionality. We observed that precision is 0.5 when Lasso is selected and higher when FBED is selected. We should stress that SES was never selected by PROTEUS for the synthetic datasets. FBED removed most of the irrelevant features leading to a predictive model with less than 10 features to approximate the decision boundary of the corresponding detector. PROTEUS$_{fs}$ achieves almost optimal recall regardless of the dimensionality and the employed detector. A slight drop in recall is observed when the precision higher than 0.8 (achieved only by FBED), while recall is optimal when Lasso is selected. Moreover, PROTEUS$_{fs}$ feature selection methods are robust to increasing data dimensionality and irrelevant feature ratio where CA-Lasso and SHAP seem to be particularly sensitive.

4.6 PROTEUS Generalization Performance

The objective of this experiment is to assess the generalization performance of PROTEUS without (PROTEUS$_{full}$) and with feature selection (PROTEUS$_{fs}$) as well as with the various feature importance alternatives, (PROTEUS$_{CA-Lasso}$, PROTEUS$_{SHAP}$, PROTEUS$_{LODA}$). Figure 6 depicts the AUC performance for each method in test set. Regarding the synthetic datasets, PROTEUS$_{fs}$ achieves very high AUC across the increasing data dimensionality with a minimum of 0.96. CA-Lasso and SHAP instead exhibit lower performances as they do not retrieve, as showed in the previous experiment, many of the relevant features. Observe that in the synthetic dataset PROTEUS$_{fs}$ generalizes better than PROTEUS$_{full}$, i.e., when using all the available features.

Regarding the real datasets, similar trends are observed with PROTEUS$_{fs}$ achieving consistently a very high generalization performance with a minimum of 0.8 in Arrhythmia in the presence of 2,570 dimensions and 90% irrelevant feature ratio. PROTEUS$_{fs}$ seems to approximate in a detector-agnostic manner, the optimal performance of LODA's feature importance method when LODA is used as the detection algorithm. This is due to the

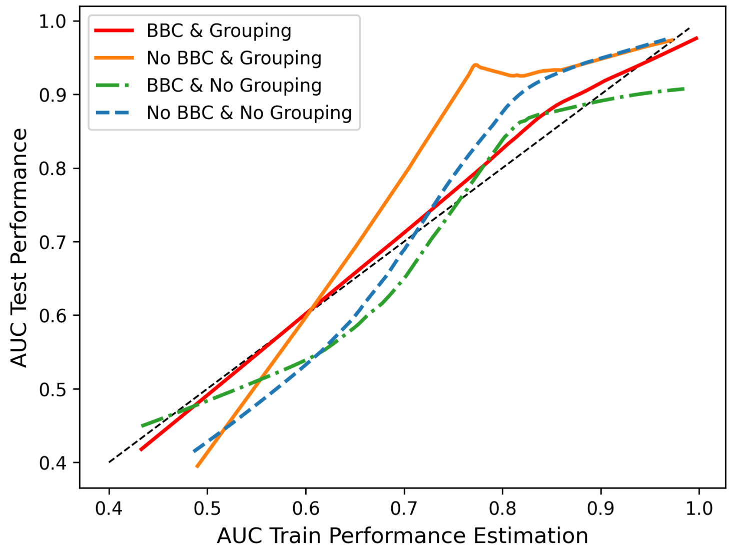


Figure 4 Bias between train and test AUC performance of PROTEUS implemented with 4 alternatives.

fact that LODA's explanations are tailored to its detection algorithm; however, if LODA's detection performance was poor in a dataset, the provided explanation would be of less value for the analysts. The feature selection methods employed by PROTEUS$_{fs}$, are able to discover the relevant features leading to predictive models with very high performance regardless of the data dimensionality (and the increasing relevant feature ratio) and capture accurately the decision boundary of every employed unsupervised detector.

4.7 PROTEUS RunTime Performance

In the subsequent experiment, we demonstrate the execution time of the feature selection methods employed by PROTEUS. Figure 7 depicts the runtime comparison between the ad-hoc feature importance methods and the feature selection algorithms. The employed methods are specifically designed to search efficiently the exponential search space and thus require less than two seconds on average in 100-dimensions to select features, exhibiting a steady execution time. In contrast, SHAP is the most expensive method; as we had to explain the outcome of any employed detector, we used Kernel SHAP which is model-agnostic. Given the fact that SHAP is optimized only for particular families of algorithms, e.g. tree-based, its execution time is particularly sensitive to data dimensionality because the Shapley values must be calculated for all the input features. Recall that in PROTEUS we tried three classifiers resulting 30 combinations according to their hyper-parameters and three feature selection algorithms resulting 20 combinations according to their hyper-parameters including the full selector, i.e., when the full feature space is considered. Thus, the total number of configurations tried in PROTEUS is 600. Each configuration requires 2 seconds on average to complete regardless of the dataset dimensionality.

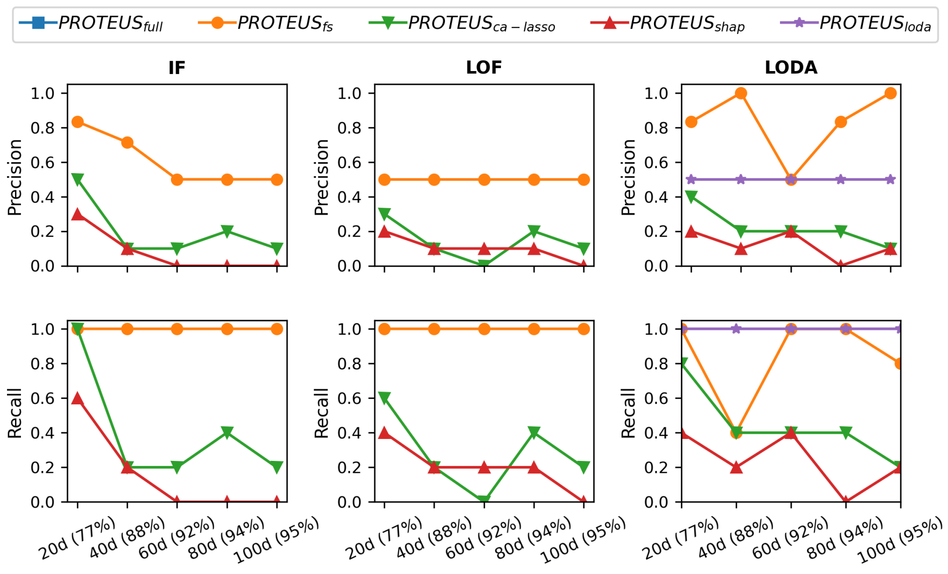


Figure 5 Precision and Recall performance of discovered features when explaining IF, LOF and LODA on synthetic datasets w.r.t. increasing data dimensionality (% irrelevant feature ratio).

5 Contrasting PROTEUS Surrogate Models with Unsupervised Anomaly Detectors

In this section, we are investigating possible disagreements between PROTEUS's surrogate models and unsupervised Anomaly Detectors, namely detected anomalies explained as normal points or vice-versa.

To assist human analysts in spotting such suspicious samples, we introduce an original visualization method based on *spider charts*. The proposed *Spider Anomaly Explanation (SAE) charts* is essentially a 2D visualisation of multivariate data projected over the explaining subspaces returned by PROTEUS. The chart has a "web-like" form with concentric circles and several spokes where each one corresponds to a specific feature. Extreme values of the features are depicted near the center or near the outermost circle. Then, a mutli-dimensional sample is represented as an irregular polygon intersecting every spoke according to the quantile its feature values falls in.

In this work, we propose a variation of spider charts tailored to anomaly explanation. First, instead of plain feature values, we consider each concentric circle in the chart to represent one of the four quantiles where the center corresponds to the 0 quantile and the outermost circle corresponds to the 1 quantile. Then, every feature value is translated to a quantile ranging from 0 to 1. Hence, the normal region in the chart is the interquartile range (IQR) containing 50% of the values. Finally, we reverse the samples with extreme low values belonging to quantiles 0 - 0.25 to the 0.75 - 1 quantiles so that both low and high extremes can be identified near the outermost circle, far from the normal region. When a sample's value intersect with a spoke in the quantiles 0.75 - 1, it means that at least 75% of values for the particular feature fall below the sample's value.

In Figure 8 we demonstrate two SAE charts when explaining the LOF detector in the Ionosphere dataset. The explanation produced by PROTEUS comprises of 9 features. In Figure 8a both PROTEUS's surrogate model and LOF agreed on the labels of these two samples. We can observe that the normal sample falls entirely into the normal (green) region

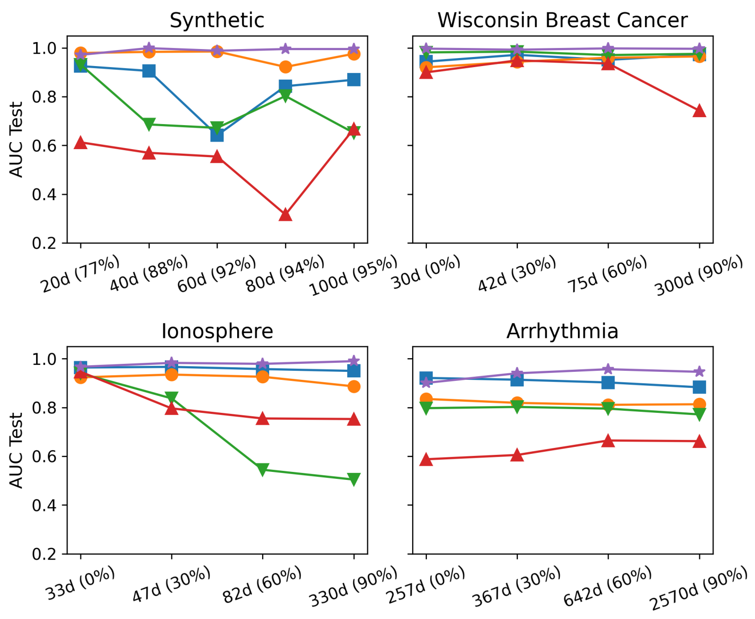


Figure 6 AUC test performance averaged over the three detectors on synthetic and real datasets w.r.t. increasing dimensionality (% irrelevant feature ratio).

while the anomalous sample deviates significantly in every feature. Figure 8b illustrates two samples where PROTEUS's surrogate model disagrees with LOF on their labels. LOF identified the blue sample as anomaly while PROTEUS identified it as normal. We can clearly observe that this sample was erroneously detected by LOF as it falls entirely into the normal region. For the other conflict (the red sample) it is not as obvious as in the former case because it deviates w.r.t. a subset of the features of the explanation. This sample is an anomaly according to the gold standard that was not detected by LOF. However, PROTEUS considered this sample an anomaly, extracting three features (Radar 10, 9 and 25) where it takes extreme values. We should finally stress that since PROTEUS strives to explain all the anomalies simultaneously, it is likely that an anomalous sample deviates w.r.t. a subset of the explaining subspace.

To quantify the utility of a PROTEUS explanation to reveal errors made by an unsupervised detector we introduce two metrics that rely on the gold standard available for each dataset. We consider as *conflicts* the suspicious samples for which the PROTEUS's surrogate model predicts a different label than the detector. Subsequently, we define two sets of conflicts following the notation of Section 2 where ω_A^l is the detector model, $f(\cdot, \theta^*)$ is the PROTEUS's surrogate model equipped with the best found hyper-parameters, "1" denotes an anomaly and "0" denotes a normal sample.

▶ **Definition 3.** Anomaly Normal Conflicts (ANC)*: Each sample that the detector model labels as anomaly while PROTEUS's surrogate model labels as normal.*

$$ANC = \{s \mid \omega_A^l(s) = 1 \wedge f(s, \theta^*) = 0\},$$

▶ **Definition 4.** Normal Anomaly Conflicts (NAC)*: Each sample that the detector model labels as normal while PROTEUS's surrogate model labels as anomaly.*

$$NAC = \{s \mid \omega_A^l(s) = 0 \wedge f(s, \theta^*) = 1\}$$

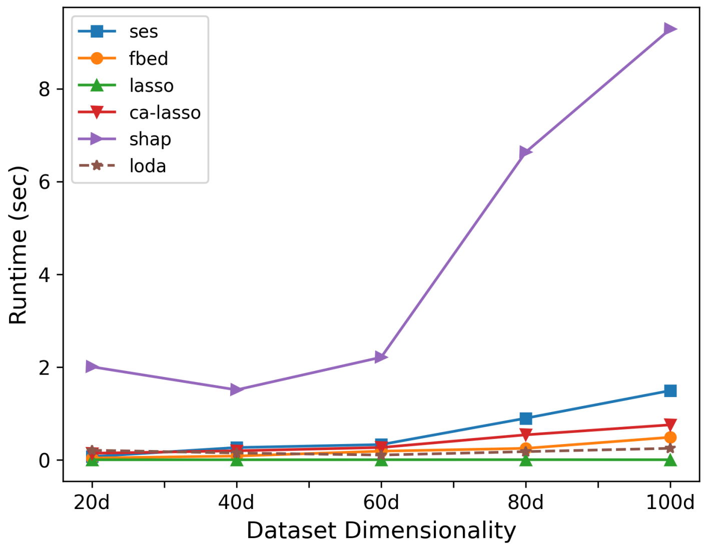


Figure 7 Average runtime of feature selection/importance methods on synthetic datasets of increasing dimensionality.

Based on the two previous sets we define two metrics to quantify the utility of a PROTEUS explanation.

▶ **Definition 5.** True Normal Discovery (TND)*: The ratio of conflicted samples that PROTEUS's surrogate model labelled correctly as normals according to the True Normals in the gold standard.*

$$TND = \frac{|ANC \cap True\ Normals|}{|ANC|}$$

▶ **Definition 6.** True Anomaly Discovery (TAD)*: The ratio of conflicted samples that PROTEUS's surrogate model labelled correctly as anomalies according to the True Anomalies in the gold standard.*

$$TAD = \frac{|NAC \cap True\ Anomalies|}{|NAC|}$$

When there are no conflicts, i.e., PROTEUS approximates perfectly the detector's decision boundary (AUC = 1 on test set), the two metrics are not defined (ANC and NAC are empty). In case of conflicts, TND and TAD range between 0 and 1. Values close to 1 indicate that PROTEUS' surrogate model disagrees with the detector model and it labels suspicious samples correctly w.r.t. the gold standard. In contrast, values close to 0 indicate that PROTEUS disagrees incorrectly with the detector. Clearly, the number of conflicts is higher when PROTEUS exhibits low AUC performance.

Figure 9a contrasts the AUC of PROTEUS against the AUC of the three detectors used to analyze each real dataset. The former is computed on the test (holdout) set using the labels produced by a detector and serves as the approximation quality of its decision boundary. The latter is computed on the train set using the labels of the gold standard and reveals the effectiveness of a detector to identify anomalies in a dataset. We can easily observe that *the quality of the approximation of a detector's decision surface by PROTEUS decreases as*

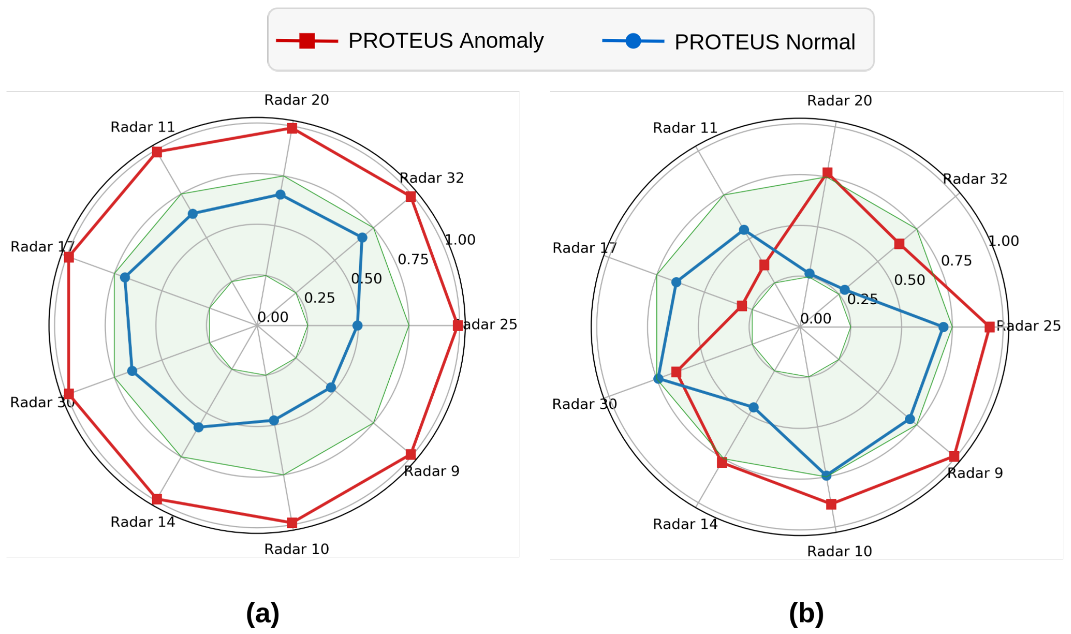
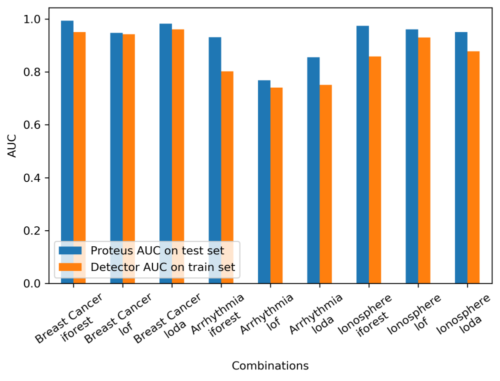


Figure 8 Spider Anomaly Explanation Charts when explaining LOF in Ionosphere using PROTEUS.

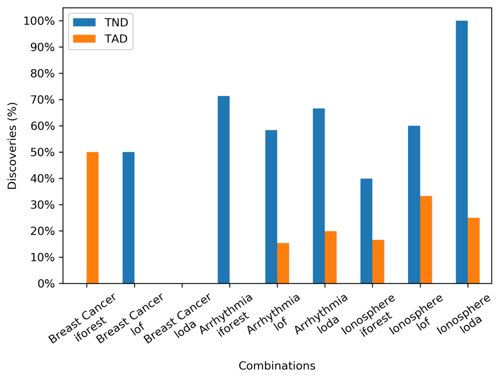
the detector's effectiveness decreases. For instance, in Arrhythmia we observe the lowest AUCs for the three detectors and also for the surrogate models of PROTEUS. This trend can be attributed to the fact that some misdetected samples are very hard to classify correctly without a very complex boundary. However, if the surrogate model needs to learn a more complex decision surface to segregate the misdetected samples from their neighbors, it makes the surrogate model prone to overfitting and thus reduces its generalization performance. Overfitting is avoided thanks to the CV protocol; PROTEUS will strive to select models that generalize well in unseen data optimizing the out-of-sample AUC performance.

Figure 9b sheds some light on the percentage of conflicting samples between PROTEUS and unsupervised detectors per dataset. PROTEUS reveals more True Normals with an average TND $\sim 50\%$ than True Anomalies with an average TAD $\sim 18\%$. In other words, *PROTEUS seems to be more effective in discovering false alarms.* To justify this claim we consider a 2D reduced visualization using t-SNE [35] of the Arrhythmia dataset projected over 10-dimensional PROTEUS explanation for LODA. Figures 10b and 10a depict the agreements between PROTEUS and LODA as circles and their disagreements as rectangles for *ANC* and triangles for *NAC*. Figure 10b illustrates the *ANC* samples contributing to the identification of *True Normals.* As expected, these samples are located within dense regions, surrounded by normal samples, requiring more complex boundaries to separate. In contrast, Figure 10a illustrates the *NAC* samples contributing to the identification of *True Anomalies.* These samples lie on sparse areas where less complex boundaries can be built to separate them. This is because less complex boundaries enable better generalizing models and thus PROTEUS can classify misdetected normals in sparse areas, not yielding many *True Anomalies.*

To conclude, PROTEUS constructs a reduced-dimensionality surrogate model that not only generalizes well to unseen samples but also provides valuable insights for identifying False Negatives and False Positives of unsupervised anomaly detectors.

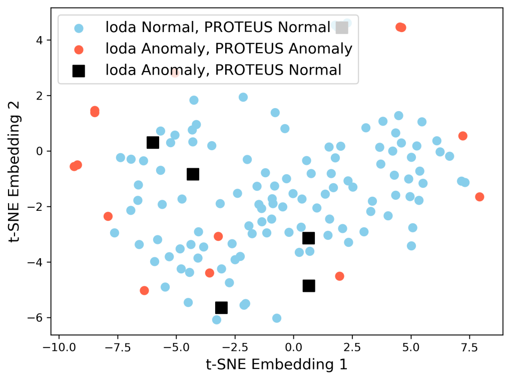


(a) AUC of anomaly detectors followed by their approximation quality (AUC) from PROTEUS for real datasets.

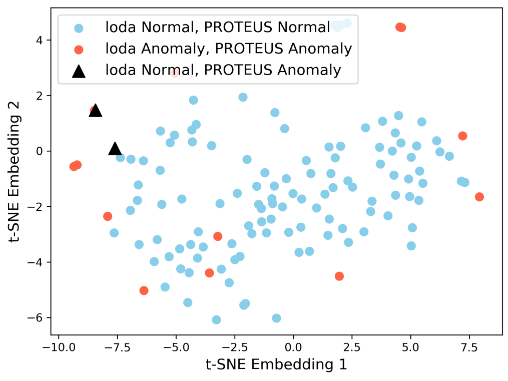


(b) Fraction of samples identified as TND and TAD according to PROTEUS explanations for different combinations of detectors and datasets.

Figure 9



(a)



(b)

Figure 10 A 2-D reduced t-SNE visualisation of Arrhythmia according to PROTEUS 10-D explanation for LODA.

6 Related Work

In this section, we survey various categories of related work on explaining anomalies in unsupervised and supervised settings, partially inspired by [36]. We should stress that explanations of anomalies in temporal data is beyond the scope of this work [16, 5].

6.1 Explainable Anomaly Detectors

As unsupervised detectors assess the abnormality of multidimensional data on various feature subspaces, they can also report the subspaces that contributed the most to the anomaly score of a particular sample. A first example of such explainable anomaly detectors is LODA [44] which scores samples based on the average log density over an ensemble of one-dimensional histogram density estimators. Given that each histogram (with sparse projections) scores a randomly generated subspace, LODA explanations are essentially a list of features ranked according to their contribution to the anomalousness score of a sample.

LODI [11] and LOGP [10] seek an optimal subspace in which an anomaly is maximally separated from its neighbors. Both works exploit a dimensionality reduction technique to measure the anomalousness of a sample in a low-dimensional subspace capable of preserving the locality around its neighbors while at the same time maximizing its distance from this neighborhood. Then, the explanation of a sample is the top-k features with the largest absolute coefficient from the eigenvector with the largest eigenvalue.

In [47], an interactive explanation method is proposed that can be used for any density-based anomaly detector. [29] introduced a method to detect anomalies in axis-parallel subspaces, called SOD, that computes the anomaly score of a in a hyperplane w.r.t. to nearest neighbors in the full space. SOD hyperplanes that contribute most in the anomaly scores serve as explanations. CMI [6] and HiCS [25] rely on statistical methods to select subspaces of high-dimensional datasets, where anomalies exhibit a high deviation from normal samples. Both consider highly contrasting subspaces as explanations of all possible anomalies in a dataset.

The previous works explain anomalies as a byproduct of an unsupervised detection method. Given that independent experimental evaluations showed that no detector outperforms all others for all possible datasets [17, 8, 13, 18], in our work we focus on learning the decision boundary of any unsupervised anomaly detector that could be used for a particular dataset. In contrast to the descriptive explanations provided by the aforementioned works, PROTEUS targets predictive explanations that could be successfully used to detect and explain anomalies also in unseen data.

6.2 Post-hoc Anomaly Explainers

The primary focus of these methods is to specify a subset of features such that a sample may obtain a high anomaly score when projected onto these subspaces. Some authors have referred to this explanation task as "outlying aspects mining" [14, 43].

We first consider works providing local explanations. The seminal work [27] first introduced the problem of explaining individual outliers with "Intensional knowledge" under the form of minimal feature subspaces in which they show the greatest deviation from inliers. To find optimal subspaces, [30] formulates a constraint programming problem that maximizes differences between neighborhood densities of known outliers and inliers. [26] employs a search strategy aiming to find a subspace which maximizes differences in anomaly score distributions of all samples across subspaces while [38] measures the separability per anomaly using classification accuracy, and then apply Lasso to produce a local explaining subspace. OAMiner [14] finds the most outlying subspace where a sample is ranked highest in terms of a probability density measure and OARank [43] ranks features based on their potential contribution toward the anomalousness of a sample. *Rather than learning the decision boundary of individual anomalies, PROTEUS builds a classifier to explain simultaneously all the anomalies spotted by an unsupervised detector. Moreover, PROTEUS's oversampling is supervised, optimizing the hyper-parameters of various feature selection and classification algorithms.*

Extending earlier work [2] on explaining individual anomalies, [3, 1] focus on explaining groups of anomalies for categorical data using contextual rule based explanations. Authors search for <context, feature> pairs, where the (single) feature can differentiate as many outliers as possible from inliers while sharing the same context. The anomalousness score of a sample in a subspace is calculated based on the frequency of the value that the outlier takes in the subspace. It tries to find subspaces E and S such that the outlier is frequent in one and much less frequent than expected in the other. To avoid searching exhaustively all such rules, the method takes two parameters, and, to constrain the frequencies of the given sample in subspaces E and S, respectively. Similarly, [56] describes anomalies grouped in time. They construct explanatory Conjunctive Normal Form rules using features with low segmentation entropy, which quantifies how intermixed normal and anomalous samples are. They heuristically discard highly correlated features from the rules to get minimal explanations. The aforementioned works assume that anomalies are scattered and strive to explain them individually rather than to summarize the explanation of a collection of anomalies.

The following works perform explanation summarization aiming to explain a set of anomalies collectively rather than individually. LookOut [19] exploits a submodular optimization function to ensure concise summarization. xPACS [36] groups anomalies by generating sequential feature-based explanations providing a ranked list of feature-value pairs that are incrementally revealed until the human expert reaches a satisfactory level of confidence. *In contrast to the interactive explanations provided by xPACS, PROTEUS provides a global feature subspace that could potentially explain even unseen anomalies.*

6.3 Explaining Black-box Predictors

Several methods have been recently proposed to explain why a supervised model predicted a particular label for a particular sample [15, 28, 40, 45]. LIME [45] constructs a linear interpretable model that is locally faithful to the predictor. In this respect, it draws uniformly at random (where the number of such draws is also uniformly sampled) pseudo-samples per every sample to be explained. Note that LIME let the black-box classifier label the generated pseudo-samples. To the best of our knowledge, LIME has not been successfully used for imbalanced neighborhoods [54]. Other works [15, 28] explain the model by perturbing the features to quantify their influence on predictions. However, these works do not aim to explain multiple examples collectively, as the global explanation problem studied in our work.

Other works aim to produce explanations in the form of feature relevance scores, which indicate the relative importance of each feature to the classification decision. Such scores have been computed by comparing the difference between a classifier's prediction score and the score when a feature is assumed to be unobserved [46], or by considering the local gradient of the classifier's prediction score with respect to the features for a particular example [4]. [48, 49] considered how to score features in a way that takes into account the joint influence of feature subsets on the classification score, which usually requires approximations due to the exponential number of such subsets.

The aforementioned works require as input a supervised model rather than an unsupervised anomaly detector. However, in real application settings it is difficult or even impossible to label data as anomalous or normal examples [17]. Moreover, *PROTEUS provides global explanations returned by standard feature selection algorithms after learning the decision boundary of the unsupervised detector.*

6.4 Evaluation of Explainers

Existing approaches for evaluating explanation methods in both supervised and unsupervised settings are typically quite limited in their scope. Often evaluations are limited to visualizations or illustrations of several example explanations [4, 10] or to test whether a computed explanation collectively conforms to some known concept in the dataset [4], often for synthetically generated data. [47] proposes a larger scale quantitative evaluation methodology for anomaly explanations regarding sequential feature explanation methods. *Compared to this study, in our work we assess the predictive performance of a classifier given an explanation along with the correctness of the learned features of the explanation.*

6.5 Imbalanced Learning

One of the main challenges in supervised anomaly detection, is class imbalance: anomalies are largely underrepresented compared to normal examples. In the following we position PROTEUS w.r.t. the main imbalanced learning methods [22]. The imbalanced learning problem is concerned with the performance of learning algorithms in the presence of under-represented data and severe class distribution skews. We follow the same categorization of imbalanced learning methods as in [22].

Random oversampling augments the original dataset by replicating examples from the minority class, while *random undersampling* removes a random set of majority class examples. PROTEUS pipelines do not perform random under/over-sampling. The synthetic minority oversampling technique (SMOTE) [9] generates new minority class examples from the line segments that join the k minority-class nearest neighbors. Our pipeline generates synthetic examples close to the original minority examples by adding gaussian noise. SVM SMOTE [42] is a SMOTE variant that generates the synthetic examples concentrated in the most critical area, i.e., the boundary discovered by fitting an SVM classifier. Borderline-SMOTE [20] seeks to oversample the minority class instances in the borderline areas, by defining a set of "Danger" examples. Adaptive Synthetic Sampling (ADASYN) [21] algorithm uses a density distribution as a criterion to automatically decide the number of synthetic examples that need to be generated for each minority example. *In comparison to the aforementioned works, PROTEUS performs a supervised synthetic minority oversampling ensuring that new samples are anomalies according to the decision boundary of an unsupervised detector that is currently explained. In addition, we proposed a method to avoid information leakage in the CV protocol when synthetic oversampling is applied.*

7 Conclusion and Future Work

We propose the first methodology for producing predictive, global anomaly explanations in a detector-agnostic fashion. In particular, we show how with adequate design choices regarding rare class oversampling and unbiased performance estimation of ML pipelines, generating predictive, global anomaly explanations boils down to an AutoML problem. As derived from our experiments, PROTEUS is not only able to discover explaining subspaces of features relevant to anomalies, but it can also construct predictive models that approximate effectively and robustly the decision boundary of popular unsupervised detectors (e.g., IF, LOF, LODA). As future work, it would be interesting to approximate the decision boundary of a detector directly from the provided anomaly scores rather than converting them to binary labels. Hence, one could transform the explanation problem into regression with feature selection.

References

1 F. Angiulli, Fabio Fassetti, L. Palopoli, and G. Manco. Outlying property detection with numerical attributes. *Data Mining and Knowledge Discovery*, 31:134–163, 2016.

2 Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Detecting outlying properties of exceptional objects. *ACM Trans. Database Syst.*, 34(1):7:1–7:62, 2009.

3 Fabrizio Angiulli, Fabio Fassetti, and Luigi Palopoli. Discovering characterizations of the behavior of anomalous subpopulations. *IEEE Trans. Knowl. Data Eng.*, 25(6):1280–1292, 2013.

4 David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, 2010.

5 Aline Bessa, Juliana Freire, Tamraparni Dasu, and Divesh Srivastava. Effective discovery of meaningful outlier relationships. *ACM/IMS Trans. Data Sci.*, 2020.

6 Klemens Böhm, Fabian Keller, Emmanuel Müller, Hoang Vu Nguyen, and Jilles Vreeken. CMI: an information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In *Proceedings of the 13th International Conference on Data Mining*, pages 198–206, 2013.

7 Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jorg Sander. Lof: identifying density-based local outliers. In *SIGMOD '00*, 2000.

8 Guilherme O. Campos, Arthur Zimek, Jörg Sander, Ricardo J. Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E. Houle. On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study. *Data Min. Knowl. Discov.*, 30:891–927, 2016.

9 Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

10 Xuan-Hong Dang, Ira Assent, Raymond T. Ng, Arthur Zimek, and Erich Schubert. Discriminative features for identifying and interpreting outliers. In *ICDE*, pages 88–99, 2014.

11 Xuan-Hong Dang, Barbora Micenková, Ira Assent, and Raymond T. Ng. Local outlier detection with interpretation. In *ECML PKDD*, pages 304–320, 2013.

12 Manuel Fernández Delgado, Eva Cernadas, Senén Barro, and Dinani Gomes Amorim. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181, 2014. `doi:10.5555/2627435.2697065`.

13 Remi Domingues, Maurizio Filippone, Pietro Michiardi, and Jihane Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2018.

14 Lei Duan, Guanting Tang, Jian Pei, James Bailey, Akiko Campbell, and Changjie Tang. Mining outlying aspects on numeric data. *Data Mining and Knowledge Discovery*, 29:1116–1151, 2014.

15 Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision, ICCV*, pages 3449–3457, 2017.

16 Ioana Giurgiu and Anika Schumann. Additive explanations for anomalies detected from multivariate temporal data. In *CIKM*, pages 2245–2248, 2019.

17 Markus Goldstein and Seiichi Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS One*, 2016.

18 Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo. Statistical analysis of nearest neighbor methods for anomaly detection. In *NeurIPS*, pages 10921–10931, 2019.

19 Nikhil Gupta, Dhivya Eswaran, Neil Shah, Leman Akoglu, and Christos Faloutsos. Beyond outlier detection: Lookout for pictorial explanation. In *ECML/PKDD*, 2018.

20 Hui Han, Wenyuan Wang, and Binghuan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *ICIC*, 2005.

21 Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. *IEEE International Joint Conference on Neural Networks*, pages 1322–1328, 2008.

22 Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *TKDE*, 21:1263–1284, 2009.

23 Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

24 David D. Jensen and Paul R. Cohen. Multiple comparisons in induction algorithms. *do. Learn.*, 38(3):309–338, 2000.

25 Fabian Keller, Emmanuel Müller, and Klemens Böhm. Hics: High contrast subspaces for density-based outlier ranking. In *ICDE*, pages 1037–1048, 2012.

26 Fabian Keller, Emmanuel Müller, Andreas Wixler, and Klemens Böhm. Flexible and adaptive subspace search for outlier analysis. In *CIKM*, pages 1381–1390, 2013.

27 Edwin M. Knorr and Raymond T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, pages 211–222, 1999.

28 Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *ICML*, volume 70, pages 1885–1894, 2017.

29 Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Outlier detection in axis-parallel subspaces of high dimensional data. In *PAKDD*, volume 5476, pages 831–838, 2009.

30 Chia-Tung Kuo and Ian Davidson. A framework for outlier description using constraint programming. In *AAAI*, pages 1237–1243, 2016.

31 Vincenzo Lagani, Giorgos Athineou, Alessio Farcomeni, Michail Tsagris, Ioannis Tsamardinos, et al. Feature selection with the r package mxm: Discovering statistically equivalent feature subsets. *Journal of Statistical Software*, 2017.

32 Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *ICDM*, 2008. `doi:10.1109/ICDM.2008.17`.

33 Scott M. Lundberg, Gabriel G. Erion, Hugh Chen, Alex J. DeGrave, Jordan M Prutkin, Bala G. Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2:56–67, 2020.

34 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017. URL: `https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html`.

35 L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

36 Meghanath Macha and Leman Akoglu. Explaining anomalies in groups with characterizing subspace rules. *Data Min. Knowl. Discov.*, 32(5):1444–1480, 2018.

37 Emaad A. Manzoor, Hemank Lamba, and Leman Akoglu. xstream: Outlier detection in feature-evolving data streams. In Yike Guo and Faisal Farooq, editors, *KDD*, pages 1963–1972, 2018. `doi:10.1145/3219819.3220107`.

38 Barbora Micenková, Raymond T. Ng, Xuan-Hong Dang, and Ira Assent. Explaining outliers by subspace separability. In *ICDM*, pages 518–527, 2013.

39 Christoph Molnar. *Interpretable Machine Learning.* independently published, 2019. URL: `https://christophm.github.io/interpretable-ml-book/`.

40 Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, 73:1–15, 2018.

41 Nikolaos Myrtakis, Ioannis Tsamardinos, and Vassilis Christophides. Proteus: Predictive explanation of anomalies. *ICDE*, 2021.

42 Hien M. Nguyen, Eric W. Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradigms*, 3:4–21, 2011.

43 Xuan Vinh Nguyen, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Jian Pei. Scalable outlying-inlying aspects discovery via feature ranking. In *PAKDD*, pages 422–434, 2015.

44 Tomás Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2015.

45 Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, 2016.

46 Marko Robnik-Sikonja and Igor Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20:589–600, 2008.

47 Md Amran Siddiqui, Alan Fern, Thomas G. Dietterich, and Weng-Keen Wong. Sequential feature explanations for anomaly detection. *ACM Trans. Knowl. Discov. Data*, 13(1):1:1–1:22, 2019.

48 Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, 2010.

49 Erik Strumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.*, 41(3):647–665, 2014.

50 Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, pages 267–288, 1996.

51 Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan E. Hines, C. Bayan Bruss, and Reza Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. In *31st IEEE International Conference on Tools with Artificial Intelligence*, pages 1471–1479, 2019.

52 Ioannis Tsamardinos, Giorgos Borboudakis, Pavlos Katsogridakis, Polyvios Pratikakis, and Vassilis Christophides. A greedy feature selection algorithm for big data of high dimensionality. *Mach. Learn.*, 108(2):149–202, 2019.

53 Ioannis Tsamardinos, Elissavet Greasidou, and Giorgos Borboudakis. Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. *Mach. Learn.*, 107(12):1895–1922, 2018. `doi:10.1007/S10994-018-5714-4`.

54 Adam White and Artur S. d'Avila Garcez. Measurable counterfactual local explanations for any classifier. In *European Conference on Artificial Intelligence, ECAI*, volume 325, pages 2529–2535, 2020. `doi:10.3233/FAIA200387`.

55 Jiawei Yang, Susanto Rahardja, and Pasi Fränti. Outlier detection: how to threshold outlier scores? In *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing, AIIPCC 2019*, pages 37:1–37:6, 2019. `doi:10.1145/3371425.3371427`.

56 Haopeng Zhang, Yanlei Diao, and Alexandra Meliou. Exstream: Explaining anomalies in event stream monitoring. In *EDBT*, pages 156–167, 2017.

On the Impact of Provenance Semiring Theory on the Design of a Provenance-Aware Database System

Pierre Senellart
DI ENS, ENS, PSL University, CNRS, Paris, France
Inria, Paris, France
Institut Universitaire de France, Paris, France
CNRS@CREATE LTD, Singapore
IPAL, CNRS, Singapore

Abstract

We report on the impact that the theory of provenance semirings, developed by Val Tannen and his collaborators, has had on the design on a practical system for maintaining the provenance of query results over a relational database, namely ProvSQL.

2012 ACM Subject Classification Theory of computation → Data provenance; Information systems → Database management system engines

Keywords and phrases provenance, provenance semiring, ProvSQL

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.9

Funding This work was funded in part by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute). It is also part of the program DesCartes and is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

Acknowledgements ProvSQL is a collective effort; I acknowledge the contributions of Yann Ramusat, Silviu Maniu, Louis Jachiet, and Baptiste Lafosse to the development of the system.

1 Introduction

The important issue of keeping track of data throughout a complex process gave rise to the study of the *provenance* [4] of data, also sometimes called *lineage* [5]: extra information attached to query results which relates them to input data items. In a seminal work in 2007 [9], Todd J. Green, Grigoris Karvounarakis, and Val Tannen put forward *provenance semirings* as an algebraic framework to express a range of different forms of provenance over relational database systems, including the *data lineage* from [5], the *why-provenance* from [4], the *Boolean provenance* implicit in the early model of incomplete information of c-tables [12] (and made explicit in [10]), and many more. This work has had a considerable impact on the understanding of what data provenance is and how it can be computed, and was largely celebrated by the research community [11]. Val Tannen, in collaboration with a number of his colleagues, then further developed the theory of provenance semirings in other works, covering topics such as compact representation of provenance for recursive queries [7], provenance of non-monotone queries [2, 6], provenance of aggregate queries [3]. This line of work was also extended to other settings than the relational one and resulted in many different applications [18].

In 2016, inspired by this beautiful theoretical framework, motivated by applications of provenance to probabilistic databases [21, 10], and frustrated by the lack of maintained software that would implement provenance semirings, we embarked on the development of

ProvSQL [19], a PostgreSQL extension that supports computation of semiring provenance and their extensions for SQL queries over relational databases. ProvSQL has first been demonstrated in 2018 [20] and has been updated and improved ever since.

In this paper, in honor of Val Tannen and the groundbreaking theory of provenance semirings, we want to reflect on the impact that theoretical research on provenance semirings has had on the design of ProvSQL: where ProvSQL directly implements the theory, where practical concerns require deviating from it, and when development is still lagging behind the theoretical framework.

We introduce semiring provenance and the way it is implemented in ProvSQL in Section 2, while in in Section 3 we discuss extensions that go beyond semiring provenance.

2 Semiring Provenance for Positive Relational Algebra Queries

We now discuss the semiring provenance framework from [9] for the positive relational algebra and how it is implemented in ProvSQL, in terms of data model, query evaluation, as well as representations of provenance expressions. We assume basic knowledge of the relational model and the relational algebra, see [1] for a primer.

2.1 Data Model

Theory

A *semiring* is an algebraic structure $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$ where $(\mathbb{K}, \oplus, \mathbb{0})$ and $(\mathbb{K}, \otimes, \mathbb{1})$ are *monoids* (a set equipped with an associative binary operation and a neutral element), $\oplus$ is commutative, $\otimes$ distributes over $\oplus$, and $\mathbb{0}$ is an absorbing element for $\otimes$ (i.e., $\forall a \in \mathbb{K}$, $a \otimes \mathbb{0} = \mathbb{0} \otimes a = \mathbb{0}$). The semiring is *commutative* if $\otimes$ is commutative. The *semiring* is often referred to by $\mathbb{K}$ when the binary operations and neutral elements are clear. Given two semirings $\mathbb{K}$ and $\mathbb{K}'$, a *semiring homomorphism* from $\mathbb{K}$ to $\mathbb{K}'$ is a function from $\mathbb{K}$ to $\mathbb{K}'$ that maps neutral elements of $\mathbb{K}$ to the corresponding neutral elements of $\mathbb{K}'$ and that preserves the binary operations of the semirings.

▶ **Example 1.** The following are classical examples of semirings, with applications to provenance:

- $(\mathbb{B} = \{\bot, \top\}, \vee, \wedge, \bot, \top)$ is the semiring of Booleans;
- $(\mathbb{N}, +, \times, 0, 1)$ is the *counting* semiring;
- $(\mathbb{S} = \{\text{unclassified} < \text{restricted} < \text{confidential} < \text{secret} < \text{top\_secret} < \text{unavailable}\}, \min, \max, \text{unavailable}, \text{unclassified}\}$ is the *security semiring* of security clearance levels;
- For any finite set X of variables, $(\mathbb{N}[X], +, \times, 0, 1)$ is the *integer polynomial semiring* that is sometimes also called *how-semiring*.

See [17] for many more examples and their application to provenance.

Given a commutative semiring $(\mathbb{K}, \oplus, \otimes, \mathbb{0}, \mathbb{1})$, [9] introduces a $\mathbb{K}$-*relation* (or *relation annotated by* $\mathbb{K}$) over a finite set of attributes A as a function R that maps tuples over A to an element of $\mathbb{K}$ such that $\{t \mid R(t) \neq \mathbb{0}\}$ is finite. Homomorphisms are extended to $\mathbb{K}$-relations: for a homomorphism $h : \mathbb{K} \to \mathbb{K}'$ and a $\mathbb{K}$-relation R, $h \circ R$ is a $\mathbb{K}'$-relation. Finally, $\mathbb{K}$-databases are databases formed of (labeled) $\mathbb{K}$-relations, and semiring homomorphisms extend to $\mathbb{K}$-databases in the natural way.

▶ **Example 2.** $\mathbb{B}$-relations over a set of attributes A are simply relations in the usual sense: finite set of tuples over A.

Table 1 Relation personnel for the personnel of an intelligence agency, used as a running example, from [17].

id	name	position	city	classification	
1	John	Director	New York	unclassified	t_1
2	Paul	Janitor	New York	restricted	t_2
3	Dave	Analyst	Paris	confidential	t_3
4	Ellen	Field agent	Berlin	secret	t_4
5	Magdalen	Double agent	Paris	top_secret	t_5
6	Nancy	HR	Paris	restricted	t_6
7	Susan	Analyst	Berlin	secret	t_7

Consider the example personnel relation in Table 1. If $t_1, \dots, t_7$ are elements of a semiring $\mathbb{K}$ distinct from $\mathbb{0}$, then this depicts a $\mathbb{K}$-relation where every tuple of the relation is associated with a non-$\mathbb{0}$ element of $\mathbb{K}$. For example, assume that for every $1 \leq i \leq 7$, t_i is set to the value of the classification attribute of the corresponding tuple; then this is a $\mathbb{S}$-relation.

The integer polynomial semiring plays an important role in the theory of provenance semirings as it is *universal* in the following sense [9, Proposition 4.2]: for any commutative semiring $\mathbb{K}$, any set X of variables, and any valuation $v : X \to \mathbb{K}$ of these variables to element of $\mathbb{K}$, there exists a unique homomorphism of semirings h from $\mathbb{N}[X]$ to K such that $h(x) = v(x)$.

▶ **Example 3.** For example, take $X = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ and the valuation v that maps each of these tuple ids to the corresponding security level from the classification attribute in Table 1. Then the unique homomorphism $h : \mathbb{N}[X] \to \mathbb{S}$ preserving this valuation is the one that maps $t_1t_2 + t_4^2t_5$ to

$$\begin{aligned} \min(\max(v(t_1), v(t_2)), \max(v(t_4), v(t_4), v(t_5))) &= \min(\max(v(t_1), v(t_2)), \max(v(t_4), v(t_5))) \\ &= \min(\text{restricted}, \text{top\_secret}) \\ &= \text{restricted}. \end{aligned}$$

Implementation in ProvSQL

ProvSQL relies on the universality of the integer polynomial semiring by representing every relation as an $\mathbb{N}[\mathcal{U}]$-relation where $\mathcal{U}$ is a set of unique identifiers of base tuples. One major difference with the theoretical framework is that relations in SQL are not sets of tuples but multisets: the definition of a $\mathbb{K}$-relation thus needs to be modified to allow multiple annotations for the same tuple, resulting in multiple occurrences of this tuple.

To create an $\mathbb{N}[\mathcal{U}]$-relation, ProvSQL provides an add_provenance function, that takes as input a regular PostgreSQL relation and modifies it to add to this relation an additional provsql attribute initialized with universally unique identifiers (UUIDs), generated at random.

When one wants to interpret such an annotated relation as a $\mathbb{K}$-relation for a semiring $\mathbb{K}$, it suffices to provide the valuation $v : \mathcal{U} \to \mathbb{K}$ (called in ProvSQL a *provenance mapping*) as a PostgreSQL relation, as well as a description of the homomorphism from $\mathbb{N}[\mathcal{U}]$ to $\mathbb{K}$, which amounts to explaining how to interpret in the semiring $\mathbb{K}$ the 0, 1 elements of $\mathbb{N}[\mathcal{U}]$, as well as the binary operations $+$ and $\times$ of $\mathbb{N}[\mathcal{U}]$. ProvSQL provides a provenance_evaluate function for this purpose; operations of the semirings are typically coded as PL/pgSQL functions, PostgreSQL's user-defined function language.

2.2 Positive Relational Algebra Query Evaluation

Theory

The same paper [9] shows how the different operators of the positive relational algebra (selection, projection, union, projection, cross product or join, and renaming) can be defined on $\mathbb{K}$-relations: selection and renaming have no effect on the annotations; tuples merged as a result of a projection or a union are combined with the $\oplus$ operation of the semiring; tuples jointly participating in producing a new tuple in a product or join are combined with the $\otimes$ operation of the semiring. Given a query q of the positive relational algebra and a $\mathbb{K}$-relation R, $q(R)$ is the $\mathbb{K}$-relation obtained by inductively applying these operations.

▶ **Example 4.** Consider the Boolean query

$$\Pi_{\emptyset}(\sigma_{\mathsf{id}<\mathsf{id2}}(\mathsf{personnel} \bowtie_{\mathsf{city}} \Pi_{\mathsf{id2,city}}(\rho_{\mathsf{id}\to\mathsf{id2}}(\mathsf{personnel}))))$$

over the running example schema which returns whether there exists a city with two distinct individuals. The result of evaluating this query over the $\mathbb{N}[X]$-relation from Table 1 is the annotated relation with a single nullary tuple annotated with the polynomial $t_1t_2+t_3t_5+t_3t_6+t_3t_6+t_4t_7$. If instead this is a $\mathbb{S}$-relation with the annotation from the classification attribute, the resulting annotation is $\min(\text{restricted}, \text{secret}, \text{top\_secret}, \text{top\_secret}, \text{secret}) = \text{restricted}$.

The reason why this is the right definition is given by two results from [9]. First, some standard identities of the relational algebra are preserved [9, Proposition 3.4]. Second, *query evaluation commutes with semiring homomorphism* [9, Proposition 3.5]: if h is a homomorphism from $\mathbb{K}$ to $\mathbb{K}'$, q a positive relational algebra query and R a $\mathbb{K}$-relation, then $h \circ q = q \circ h$.

Implementation in ProvSQL

Again, the theory needs to be adapted to reflect the fact that SQL uses a multiset semantics and not a set semantics. This has an impact for projections (which does not imply duplicate eliminations in SQL) and for `UNION ALL` unions: they do not change the provenance annotations. On the other hand, `DISTINCT` and `GROUP BY` operators in SQL result in duplicate elimination and thus in the application of the $\oplus$ operator of the semiring.

All operations are done in the $\mathbb{N}[\mathcal{U}]$ semiring. Because of the commutativity of semiring homomorphisms and query evaluation, it is possible to evaluate the result of a query in a different semiring by first evaluating it in the $\mathbb{N}[\mathcal{U}]$ semiring and then apply the semiring homomorphism.

In ProvSQL, the $\oplus$ and $\otimes$ operations of the semiring are respectively implemented by a provenance_plus and provenance_times user-defined function. At planning time, the query sent to PostgreSQL is rewritten so that the resulting relation includes a provsql column whose content is computed using these two functions, following the operations specified in the query.

▶ **Example 5.** Consider the following SQL query, which uses the usual `SELECT DISTINCT 1` trick to mimic the behavior of the Boolean query from Example 4:

```
SELECT DISTINCT 1 FROM (
    SELECT p1.city
    FROM personnel p1
    JOIN personnel p2 ON p1.city=p2.city
    WHERE p1.id<p2.id
    GROUP BY p1.city
  ) inner_query;
```

In ProvSQL, this query gets rewritten to the following one so as to produce in a new provsql attribute the correct provenance annotation: provenance_times gets called to reflect the join, while provenance_plus gets called to reflect both the `GROUP BY` and `DISTINCT` operators (the latter being converted to a `GROUP BY`).

```
SELECT 1, provenance_plus(ARRAY_AGG(provsql)) AS provsql FROM (
  SELECT p1.city, provenance_plus(
    ARRAY_AGG(provenance_times(p1.provsql,p2.provsql))) AS provsql
  FROM personnel p1 JOIN personnel p2 ON p1.city=p2.city
  WHERE p1.id<p2.id
  GROUP BY p1.city
) inner_query
GROUP BY 1;
```

Another challenge of the practical implementation is that PostgreSQL's internal data structures do not fully match the abstract view of the relational algebra; instead, every operator that exists in the SQL language gets reflected in a special way, which requires handling many subcases (and which means SQL support in ProvSQL is still not complete to this date).

2.3 Provenance Representations

Theory

Though [9] does not explictly give complexity results, it is clear that provenance tracking can be done in polynomial-time. The exact complexity, however, depends on how costly the $\oplus$ and $\otimes$ operations of the semiring are. If they can be reasonably counted to be in $O(1)$ for certain application semirings (e.g., the security semiring or even the counting semiring if integers involved are bounded), one needs to be more careful about the complexity of operations in more complex semirings such as the integer polynomial semiring. Indeed, if one were to require expanding every polynomial to a sum of monomial, it is easy to construct examples where this results in exponentially-sized expressions.

The question of compact representation of provenance led Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen to propose in [7] *arithmetic circuits* to represent provenance annotations in a way that allows sharing and does not require copying entire subexpressions or expanding them. This was done in the context of recursive queries (see 3.1) but is also useful for non-recursive ones.

Implementation in ProvSQL

Since all provenance in ProvSQL is $\mathbb{N}[\mathcal{U}]$ provenance, compact representation is paramount for efficiency of query evaluation as well as reduced use of storage. ProvSQL thus stores provenance as an arithmetic circuit, whose internal gates are the semiring operators and leaves are base UUIDs. This way, the provsql column of provenance-aware relations can simply be pointers to the corresponding gates in the circuit (in practice, we also use UUIDs as identifiers of internal gates, and these UUIDs are stored in the provsql columns). The provenance_plus and provenance_times functions add new gates to the circuit. This raises the question of where to store the circuit. We have successively experimented with three different storage mechanisms:

1. Initially, the provenance circuit was stored as a table within the same database, managed by the database engine. Unfortunately, this is extremely inefficient, as this means that every query results in many different updates (each time a gate is created in the circuit) on the provenance circuit table; this was also a nightmare in terms of concurrency control as every query turned into a batch of updates.
2. We then moved to storing the circuit in main memory, using the shared memory buffers of PostgreSQL. This was much more efficient and made it easier to address concurrency issues, but this was not a viable solution either, as the amount of shared memory buffers is limited, and this solution does not provide any way to ensure persistence of storage of the circuit.
3. In our latest implementation, the circuit is stored on disk, in memory-mapped files that are accessed through a single process. This solution resolves the issues of persistence and concurrency control, while memory mapping helps with keeping access to the circuit efficient in practice.

3 Beyond Positive Relational Algebra Queries and Semirings

We now briefly discuss theoretical ways that have been proposed to go beyond the positive relational algebra and the semiring frameworks, and their influence in the design of ProvSQL.

3.1 Recursive Queries

Theory

The original paper on the semiring framework [9] also dealt with recursive queries in the form of Datalog programs. Green, Karvounarakis, and Tannen showed that their provenance could be captured by semirings, as long as those satisfied some technical conditions (in particular, being ω-continuous). In this setting, most of the results for the positive relational algebra can be recovered: commutativity of Datalog queries and semiring homomorphisms, as well as the existence of a universal semiring, i.e., the semiring of formal power series. Algorithms for computing the provenance of recursive queries were then refined in [7] with the introduction of provenance circuits. Recent work by other authors [14] study in more detail conditions for convergence of Datalog queries involving provenance.

Implementation in ProvSQL

Unfortunately, support for recursive queries cannot be added to ProvSQL in a straightforward way. This is due to the fact that the computation of the provenance annotation in the special provsql columns requires aggregation to combine annotations of different tuples, and that SQL forbids aggregation within `WITH RECURSIVE` recursive queries. There does not seem to be any easy way around this without reimplementing a query evaluation engine, which is out of the scope of the ProvSQL project. Note, however, that together with Yann Ramusat and Silviu Maniu, we have proposed and experimented various algorithms for evaluation of the provenance of recursive queries [15, 16], inspired by [9, 7], but in a simpler setting outside of a database engine.

3.2 Non-Monotone Queries

Theory

A natural direction beyond the positive relational algebra is to add negation, by adding the difference operator of the full relational algebra. One way to add them to the framework of provenance semirings is to extend semirings with a *monus* $\ominus$ operator (which results in what is called *m-semirings*), as proposed by Floris Geerts and Antonella Poggi [8]; for example, in the Boolean semiring it is as expected defined as $a \ominus b = a \wedge \neg b$ and in the counting semiring as $a \ominus b = \max(0, a - b)$. However, Yael Amsterdamer, Daniel Deutch, Tova Milo, and Val Tannen have showed in [2] that this definition results in some counter-intuitive results (some common axioms, such as distributivity over $\otimes$ over $\ominus$, fail); in addition, $\mathbb{N}[X]$ is not a universal semiring with monus [8].

As an alternative to semirings with monus, Katrin M. Dannert, Erich Grädel, Matthias Naaf, and Val Tannen have proposed [6] a very general logical framework for computing the provenance of recursive queries (in the form of fixpoint logics) with negation. This is based on a trick of associating with every positive provenance token a corresponding negative one and considering the semiring of integer polynomials (or formal series in the recursive case) with variables both positive and negative tokens.

A final alternative for provenance with difference is given by the work on provenance aggregate [3] that we discuss in the next section.

Implementation in ProvSQL

ProvSQL follows the m-semiring approach, despite its limitations identified in [2]. Since $\mathbb{N}[\mathcal{U}]$ is not universal any longer, we need to work with the actual universal m-semiring, which is simply the free m-semiring [8], i.e., the m-semiring of free terms constructed using $\oplus$, $\otimes$, $\ominus$, quotiented by the equivalence relations imposed by the m-semiring structure. In practice, this means adding a `provenance_monus` function, used when the `EXCEPT` SQL keyword is used, that adds a $\ominus$ gate in the provenance circuit.

3.3 Aggregate Queries

Theory

Another very commonly used query feature that goes beyond the relational algebra is *aggregates*. Yael Amsterdamer, Daniel Deutch, and Val Tannen have proposed a solution [3] in the form of *provenance semimodules* for the case of aggregate functions that are associative and commutative, such as min, max, `sum`, or `count`. The scalar aggregate values form a monoid, which is combined with the provenance semiring to form a semimodule. Note that the resulting semimodule values are now annotating attribute values instead of annotating tuples. In addition to these semimodule attribute values, [3] introduces an additional δ operator to the provenance semiring that is used to determine the tuple annotation of tuples that include an aggregate computation (see [3]). Finally, when selection can be done on the result on an aggregation, additional comparison operators are introduced to build provenance annotations (these operators can then be used to define a semantics for difference).

▶ **Example 6.** Consider the query that counts the number of distinct cities in the running example schema. When evaluating this query over the $\mathbb{N}[X]$-relation from Table 1 we obtain a unary tuple with semimodule value

$$(t_1 \oplus t_2) \star 1 + (t_3 \oplus t_5 \oplus t_6) \star 1 + (t_4 \oplus t_7) \star 1$$

where $\star$ is a tensor product allowing combining elements of the aggregation monoid with elements of the provenance semiring and $+$ is the aggregation monoid operation (here, addition). The provenance annotation of this tuple is 1 (the $\mathbb{1}$-element of the semiring) as it is always present. In the security semiring, this is:

$$\text{unclassified} \star 1 + \text{confidential} \star 1 + \text{secret} \star 1$$

with provenance annotation "unclassified".

Implementation in ProvSQL

ProvSQL carefully follows the theoretical framework of [3] to support provenance computation of aggregate queries. At the moment, aggregates are only supported when they are the final operation performed, though we have plans of adding support of nested aggregates in the future. In order to keep a compact representation of the aggregate values, these are also added to the provenance circuit, using extra gate types for the tensor product and monoid aggregate operators.

4 Conclusion

In this paper, we have presented the tremendous impact that the work of Val Tannen and his collaborators on provenance semirings has had on the design of a practical system for computing the provenance of query results. It is remarkable that so many of these theoretical works lead themselves to practical implementations that can be made efficient.

Though ProvSQL is already usable as is (and, in addition to provenance, provides features for computations of probabilities [17] and Shapley(-like) values [13]), there remains a number of features to implement and optimizations to perform. The most direct given the previous discussion is the support for nested aggregates; recursive queries are unfortunately unlikely to be supported in a near future. Performing all computations in the universal semiring (or the universal m-semiring) has the advantage of being a generic approach, but means that many optimizations that are possible in a given semiring cannot be applied – some engineering is required to allow a user to request ProvSQL to capture only specific forms of provenance and ensure all possible optimizations for this particular algebraic structure are performed.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

2 Yael Amsterdamer, Daniel Deutch, and Val Tannen. On the limitations of provenance for queries with difference. In Peter Buneman and Juliana Freire, editors, *3rd Workshop on the Theory and Practice of Provenance, TaPP'11, Heraklion, Crete, Greece, June 20-21, 2011*. USENIX Association, 2011. URL: `https://www.usenix.org/conference/tapp11/limitations-provenance-queries-difference`.

3 Yael Amsterdamer, Daniel Deutch, and Val Tannen. Provenance for aggregate queries. In Maurizio Lenzerini and Thomas Schwentick, editors, *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 153–164. ACM, 2011. `doi:10.1145/1989284.1989302`.

4 Peter Buneman, Sanjeev Khanna, and Wang Chiew Tan. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330. Springer, 2001. `doi:10.1007/3-540-44503-X_20`.

5 Yingwei Cui and Jennifer Widom. Practical lineage tracing in data warehouses. In David B. Lomet and Gerhard Weikum, editors, *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 - March 3, 2000*, pages 367–378. IEEE Computer Society, 2000. `doi:10.1109/ICDE.2000.839437`.

6 Katrin M. Dannert, Erich Grädel, Matthias Naaf, and Val Tannen. Semiring provenance for fixed-point logic. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 17:1–17:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.CSL.2021.17`.

7 Daniel Deutch, Tova Milo, Sudeepa Roy, and Val Tannen. Circuits for Datalog provenance. In Nicole Schweikardt, Vassilis Christophides, and Vincent Leroy, editors, *Proc. 17th International Conference on Database Theory (ICDT), Athens, Greece, March 24-28, 2014*, pages 201–212. OpenProceedings.org, 2014. `doi:10.5441/002/icdt.2014.22`.

8 Floris Geerts and Antonella Poggi. On database query languages for k-relations. *J. Appl. Log.*, 8(2):173–185, 2010. `doi:10.1016/j.jal.2009.09.001`.

9 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In Leonid Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM, 2007. `doi:10.1145/1265530.1265535`.

10 Todd J. Green and Val Tannen. Models for incomplete and probabilistic information. In Torsten Grust, Hagen Höpfner, Arantza Illarramendi, Stefan Jablonski, Marco Mesiti, Sascha Müller, Paula-Lavinia Patranjan, Kai-Uwe Sattler, Myra Spiliopoulou, and Jef Wijsen, editors, *Current Trends in Database Technology - EDBT 2006, EDBT 2006 Workshops PhD, DataX, IIDB, IIHA, ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Munich, Germany, March 26-31, 2006, Revised Selected Papers*, volume 4254 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2006. `doi:10.1007/11896548_24`.

11 Todd J. Green and Val Tannen. The semiring framework for database provenance. In Emanuel Sallinger, Jan Van den Bussche, and Floris Geerts, editors, *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 93–99. ACM, 2017. `doi:10.1145/3034786.3056125`.

12 Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984. `doi:10.1145/1634.1886`.

13 Pratik Karmakar, Mikaël Monet, Pierre Senellart, and Stéphane Bressan. Expected Shapley-like scores of boolean functions: Complexity and applications to probabilistic databases. *Proc. ACM Manag. Data*, 2(2 (PODS)), 2024. `doi:10.1145/3651593`.

14 Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, Dan Suciu, and Yisu Remy Wang. Convergence of datalog over (pre-) semirings. *SIGMOD Rec.*, 52(1):75–82, 2023. `doi:10.1145/3604437.3604454`.

15 Yann Ramusat, Silviu Maniu, and Pierre Senellart. Provenance-based algorithms for rich queries over graph databases. In Yannis Velegrakis, Demetris Zeinalipour-Yazti, Panos K. Chrysanthis, and Francesco Guerra, editors, *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 73–84. OpenProceedings.org, 2021. `doi:10.5441/002/EDBT.2021.08`.

16 Yann Ramusat, Silviu Maniu, and Pierre Senellart. Efficient provenance-aware querying of graph databases with datalog. In Vasiliki Kalavri and Semih Salihoglu, editors, *GRADES-NDA '22: Proceedings of the 5th ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA), Philadelphia, Pennsylvania, USA, 12 June 2022*, pages 4:1–4:9. ACM, 2022. doi:10.1145/3534540.3534689.

17 Pierre Senellart. Provenance and probabilities in relational databases. *SIGMOD Rec.*, 46(4):5–15, 2017. doi:10.1145/3186549.3186551.

18 Pierre Senellart. Provenance in databases: Principles and applications. In Markus Krötzsch and Daria Stepanova, editors, *Reasoning Web. Explainable Artificial Intelligence - 15th International Summer School 2019, Bolzano, Italy, September 20-24, 2019, Tutorial Lectures*, volume 11810 of *Lecture Notes in Computer Science*, pages 104–109. Springer, 2019. doi:10.1007/978-3-030-31423-1_3.

19 Pierre Senellart. ProvSQL. https://github.com/PierreSenellart/provsql, 2024.

20 Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. ProvSQL: Provenance and probability management in PostgreSQL. *Proc. VLDB Endow.*, 11(12):2034–2037, 2018. doi:10.14778/3229863.3236253.

21 Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011. doi:10.2200/S00362ED1V01Y201105DTM016.

Different Differences in Semirings

Dan Suciu
University of Washington, Seattle, WA, USA

To Val Tannen, my teacher, mentor, and friend

Abstract

Relational algebra operates over relations under either set semantics or bag semantics. In 2007 Val Tannen extended the semantics of relational algebra to K-relations, where each tuple is annotated with a value from a semiring. However, only the positive fragment of the relational algebra can be interpreted over K-relations. The reason is that a semiring contains only the operations *addition* and *multiplication*, and does not have a difference operation. This paper explores three ways of adding a difference operator to a semiring: as a freely generated algebra, by using the natural order, or by an explicit construction using products and quotients. The paper consists of both a survey of results from the literature, and of some novel results.

2012 ACM Subject Classification Information systems → Structured Query Language

Keywords and phrases Semirings, K-relations

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.10

Funding NSF-BSF 2109922, NSF IIS 2314527, NSF SHF 2312195.
Dan Suciu: Partially Supported by NSF IIS 2314527.

1 Introduction

Val Tannen's seminal paper [13] extended the positive relational algebra to *K-relations*, where each tuple of the relation is associated with an element of the semiring. Yet this elegant generalization excluded one operator: set difference. The reason is that a semiring defines only the $\oplus$ and $\otimes$ operators and there is no canonical way to add a minus operation, although some semirings appear to admit a natural difference operator, see examples in Sec. 2. This lead several researchers to propose ways to define minus on K-relations. Tannen considered using the *ring* $\mathbb{Z}$ instead of a semiring in [12], and proved that relational algebra expressions admit a canonical form, as sum-of-product expressions. In recent work [8, 11] Tannen used dual-indeterminate polynomials, $\mathbb{N}[X, \bar{X}]$, where a positive and negative variable interact via the identity $x \cdot \bar{x} = 0$.

In this paper we explore three alternative ways to define a difference operator in a semiring; the paper consists both of a survey of related work, and some novel contributions. The first and most obvious approach to define difference algebraically. For any set of desired identities there is a unique way to extend freely a semiring with a difference operation that satisfies those identities. This is a standard technique in universal algebras and we review it in Sec. 3. The question is, what set of identities we should choose. For example if we ask for difference to convert the semiring into a ring, then the freely generate ring may collapse to a trivial ring. For example, the ring freely generated by the natural numbers $\mathbb{N}$ is $\mathbb{Z}$, but the ring freely generated by the Booleans $\mathbb{B}$ is the trivial ring $\{0\}$.

Therefore, in Section 4, we explore an alternative way to define difference: assuming that the semiring is naturally ordered, one can define the difference $b \ominus a$ as the smallest element z such that $a \oplus z \succeq b$. Bosbach [5] and Amer [3] considered naturally ordered semigroups, and monoids respectively, where such a difference operation exists, and proved that they form an equational class that can be described by a small set of axioms. This result is quite surprising, because the natural order does not appear to have a clear algebraic definition: we review this result and present a short, self-contained proof in Sec. 4.2. Geerts and Poggi [9] introduced m-semirings, which are naturally ordered semirings where the difference operation exists. Tannen [4] proved that m-semirings fail to satisfy an important axiom (called $(A5)$ in this paper) that is needed in query optimization: we review this in Sec. 4.3. Reference [4] ends by suggesting the addition of the axiom $(A5)$ to those of an m-semiring, in order to ensure that current optimizations performed by a query optimizer continue to hold over K-relations. However, adding $(A5)$ turns out to be insufficient. We show that, in order to preserve all identities valid under bag semantics, one must ensure that the semiring satisfies all identities that hold in $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$, where $\dot{-}$ is *monus*, an operation defined below in Eq. (3): we prove in Appendix A that this set is co-r.e. complete, and, thus, it is not finitely axiomatizable.

The definition of monus in Section 4 requires the semiring to be naturally ordered. An interesting question is whether the naturally ordered semirings form an equational class, i.e. whether they can be described by a set of identities. We answer this in Sec. 4.4: while natural order is not definable by a set of equations, it becomes definable if one allows one additional auxiliary operator.

Finally, in Sec. 5 we discuss a third, constructive method for adding difference, by following the same methodology as in the construction of integers $\mathbb{Z}$ from natural numbers $\mathbb{N}$. The traditional construction consists of equivalence classes of pairs (x, y) of natural numbers, where the equivalence relation is given by $(x, y) \equiv (u, v)$ when $x + v = y + u$. This set is isomorphic to $\mathbb{Z}$, and is often taken as the definition of $\mathbb{Z}$. We study whether this definition can be generalized from $\mathbb{N}$ to semirings. If we use the congruence $\equiv$ above, then the quotient semiring is often a trivial semiring, so we look for weaker notions of $\equiv$. We review the concept of an ideal I in a semiring in Sec. 5.2. In a ring, any ideal I defines a congruence relation $x \equiv_I y$, as $x - y \in I$. We describe two alternative ways to define $\equiv_I$ in a semiring, and use them to generalize the $\mathbb{N}$-to-$\mathbb{Z}$ construction. When applied to Booleans $\mathbb{B}$, or to a Boolean algebra, or to the tropical semiring Trop, this produces an extended semiring with a difference operation, which contains *positive elements* x, *negative elements* $\bar{x}$, and *over determined elements* of the form $x + \bar{x}$. These semirings resemble somewhat the dual-indeterminate polynomials $\mathbb{N}[X, \bar{X}]$ introduced in [8,11], yet they are quite different, for example they do not satisfy the identity $x\bar{x} = 0$.

Finally, we conclude with a short discussion in Sec. 6.

2 Problem Definition and Examples

A *semiring* is a tuple $\boldsymbol{S} = (S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, where $(S, \oplus, \mathbf{0})$ is a commutative monoid, $(S, \otimes, \mathbf{1})$ is a monoid, $\otimes$ distributes over $\oplus$, and $x \otimes \mathbf{0} = \mathbf{0} \otimes x = \mathbf{0}$. When $\otimes$ is also commutative then we say that the semiring is commutative. We only consider commutative semirings in this paper. When no confusion arises we will denote the operators with $+, \cdot$, and the identities with $0, 1$, without boldface. The semiring is *trivial* when $0 = 1$: in that case $S = \{0\}$, because $x = x \cdot 1 = x \cdot 0 = 0$ for all x.

We denote by Σ_m and Σ_s the signature[1] of monoids and semirings, and by Σ_{mm}, Σ_{sm} their extension with a minus operator, thus:

$$\text{Monoids:} \qquad \Sigma_m \overset{\text{def}}{=} \{+, 0\} \qquad \Sigma_{mm} \overset{\text{def}}{=} \Sigma_m \cup \{-\} \tag{1}$$

$$\text{Semirings:} \qquad \Sigma_s \overset{\text{def}}{=} \{+, \cdot, 0, 1\} \qquad \Sigma_{sm} \overset{\text{def}}{=} \Sigma_s \cup \{-\} \tag{2}$$

[1] A signature is also called a vocabulary, or a type.

where $+, \cdot, -$ have arity 2, and $0, 1$ have arity 0. Thus, every monoid is an Σ_m-algebra, and every semiring is a Σ_s-algebra. The problem discussed in this paper is to extend an arbitrary semiring from an Σ_s-algebra to an Σ_{sm}-algebra; some of the discussion will be focused on how to extend the additive monoid to an Σ_{mm}-algebra.

Many semirings already admit a natural difference operator. We illustrate with some examples.

- Every *ring* is a semiring where, for each element x, there exists some $-x$, called the *inverse*, such that $x + (-x) = 0$. The inverse is unique, and $a \cdot (-x) = -(a \cdot x)$. Any ring can be naturally extended to an Σ_{sm}-algebra by defining minus as $y - x \stackrel{\text{def}}{=} y + (-x)$.
- The semiring of natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$ can be also extended to an Σ_{sm}-algebra, by defining the *monus* operation as:

$$y \dot{-} x \stackrel{\text{def}}{=} \begin{cases} y - x & \text{when } y \geq x \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

- The semiring of Booleans $\mathbb{B} = (\{0, 1\}, \vee, \wedge, 0, 1)$ can be extended with the minus operator $y - x \stackrel{\text{def}}{=} y \wedge (\neg x)$. This extends to any Boolean algebra $(2^\Omega, \cup, \cap, \emptyset, \Omega)$ by defining difference as the standard set difference $y \setminus x$.
- An interesting example is the tropical semiring, $\mathsf{Trop} = ([0, \infty], \min, +, \infty, 0)$, where difference can be defined as:

$$b \ominus a \stackrel{\text{def}}{=} \begin{cases} \infty & \text{when } b \geq a \\ b & \text{otherwise} \end{cases} \tag{4}$$

  This operator is used for semi-naive evaluation of datalog programs over the tropical semiring [1]. For example, consider the APSP (All Pairs Shortest Path) problem. If the current shortest distance between two nodes is $d[x, y] = a$, and the algorithm discovers a new path of length b, then it updates $d[x, y] := \min(a, b)$. The semi-naive algorithm optimizes this step by first computing the difference $\delta \stackrel{\text{def}}{=} b \ominus a$ (using (4)), then updating $d[x, y] := \min(a, \delta)$, which the reader can verify is equal to $\min(a, b)$. The advantage is that, when $b \geq a$ then $\delta = \infty$ and no update is necessary: the algorithm simply ignores all edges where $\delta = \infty$, resulting in a smaller join between the edge relation and the δ relation.

These simple examples don't seem to have a unifying theme. Given an arbitrary semiring $\boldsymbol{S}$, what is the natural way to define difference? We discuss in this paper three approaches to define difference.

3 Difference by Equations

The first approach is to choose a set of identities E that we want the difference operator to satisfy, then consider the Σ_{sm}, E-algebra *freely generated* by $\boldsymbol{S}$. To explain this, we need a quick review universal algebras; there are many good textbooks, for example [6] is available online.

Given a signature Σ, a Σ-algebra is a pair $\boldsymbol{A} = (A, F)$ where F is a set of functions $f^A : A^n \to A$, one for each symbol $f \in \Sigma$ of arity n. Homomorphisms between Σ-algebras, $h : \boldsymbol{A} \to \boldsymbol{B}$, are defined in a straightforward way. The *free algebra* generated by a set X, denoted $T_\Sigma(X)$, is the set of all terms that can be formed from variables in X and function symbols in Σ, see e.g. [6].

An *identity* is a pair $(e_1, e_2) \in T_\Sigma(X)$. If E is a set of identities, then an Σ, E-algebra is an algebra that satisfies[2] all identities in E. The class of all Σ, E-algebras is called an *equational class*, or a *variety.*

A powerful tool for defining difference is the following theorem:

▶ **Theorem 1.** *Let $\Sigma_0 \subseteq \Sigma$, and let E be a set of Σ-identities. Then for any Σ_0-algebra $\boldsymbol{A}$, there exists a pair $(T_{\Sigma,E}(\boldsymbol{A}), \eta)$, where $T_{\Sigma,E}(\boldsymbol{A})$ is a Σ-algebra, $\eta : \boldsymbol{A} \to T_{\Sigma,E}(\boldsymbol{A})$ is a Σ_0-homomorphism, and the following property holds. For every Σ, E-algebra $\boldsymbol{B}$, and any Σ_0-homomorphism $h : \boldsymbol{A} \to \boldsymbol{B}$, there exists a unique Σ-homomorphism $\bar{h}$ such that the following diagram commutes:*

$$\begin{tikzcd}
\boldsymbol{A} \arrow[r, "\eta"] \arrow[dr, "h"'] & T_{\Sigma,E}(\boldsymbol{A}) \arrow[d, dotted, "\bar{h}"'] \\
 & \boldsymbol{B}
\end{tikzcd} \tag{5}$$

$T_{\Sigma,E}(\boldsymbol{A})$ is unique up to homomorphism, is called the Σ, E-algebra freely generated *by $\boldsymbol{A}$, and the diagram above is called the* universality property *of $T_{\Sigma,E}(\boldsymbol{A})$.*

This is a very powerful theorem. It says that we can always add new operators to Σ_0, and enforce new identities, in a canonical way. While the proof of the theorem is constructive[3], it is not practical. $T_{\Sigma,E}(\boldsymbol{A})$ may be a superset of $\boldsymbol{A}$, or may be a homomorphic image, or may simply collapse to a trivial algebra with a single element. The theorem only tells us that $T_{\Sigma,E}(\boldsymbol{A})$ exists and is unique. We can use the theorem to add a difference operation to any semiring: all we need is to choose what identities we want difference to satisfy. For example, assume we choose the ring identities:[4]

$$x - x = \mathbf{0} \tag{6}$$

$$x + (y - z) = (x + y) - z \tag{7}$$

Then $T_{\Sigma_s,E}(\boldsymbol{S})$ is the ring freely generated by the semiring $\boldsymbol{S}$. For example, if we apply this construction to the natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$, then $T_{\Sigma_{sm},E}(\mathbb{N})$ is isomorphic to $\mathbb{Z}$. But the freely generated ring can sometimes be trivial, as can seen from this lemma.

▶ **Lemma 2.** *Let $\boldsymbol{S}$ be a semiring where addition is idempotent, $x + x = x$. Then, if $\boldsymbol{R}$ is any ring such that there exists a homomorphism $h : \boldsymbol{S} \to \boldsymbol{R}$, then $\boldsymbol{R}$ is the trivial ring. In particular, the ring freely generated by $\boldsymbol{S}$ is trivial.*

Proof. In $\boldsymbol{S}$ it holds that $1 + 1 = 1$, therefore $1 + 1 = h(1) + h(1) = h(1) = 1$ holds in $\boldsymbol{R}$. By adding -1 to both sides we obtain $0 = 1$ in $\boldsymbol{R}$, hence $\boldsymbol{R}$ is trivial. ◀

The take-away is that, in order to define a difference operation "freely", we need to choose carefully what identities we want it to satisfy. If we insist on the ring identities, then we may end up with the trivial ring. Yet, Sec. 2 showed useful examples of difference operations that were not rings. We consider next an alternative way to defined difference, by using the natural order.

---

[2] For a formal definition of what it means for an algebra $\boldsymbol{A}$ to satisfy (e_1, e_2) we refer to [6].

[3] $T_{\Sigma,E}(\boldsymbol{A})$ is defined as $T_{\Sigma \cup A}/ \equiv_{E \cup E_{\boldsymbol{A}}}$, where $\Sigma \cup A$ extends Σ with one nulary operator a for every constant $a \in A$, the set $E_{\boldsymbol{A}}$ consists of all grounded identities of the form $(f(a_1, \ldots, a_m), b)$ where $b = f^A(a_1, \ldots, a_m)$, and $\equiv_{E \cup E_{\boldsymbol{A}}}$ is the smallest congruence relation that contains $E_{\boldsymbol{A}}$ and all groundings of E

[4] Equations (6) and (7) imply that the operation $-x \overset{\text{def}}{=} \mathbf{0} - x$ is the additive inverse, because $x + (-x) = x + (\mathbf{0} - x) = (x + \mathbf{0}) - x = x - x = \mathbf{0}$.

4 Difference by Natural Order

Given a commutative monoid $\boldsymbol{M} = (M, +, 0)$ the *natural preorder*, $\preceq$, is defined as follows:

$$a \preceq b \quad \text{if} \quad \exists z : a + z = b \tag{8}$$

Then, $\preceq$ is transitive and reflexive, thus a preorder. When it is antisymmetric then it is called the *natural order* of $\boldsymbol{M}$, and $\boldsymbol{M}$ is called a naturally ordered monoid. In a naturally ordered monoid 0 is the smallest element: $0 \preceq x$ for all $x \in M$. For example, $(\mathbb{N}, +, 0)$ is naturally ordered, while $(\mathbb{Z}, +, 0)$ is not. Similarly, a *naturally ordered semiring* is a semiring $(S, +, \cdot, 0, 1)$ where the additive monoid $(S, +, 0)$ is naturally ordered. Many semirings are naturally ordered, so it makes sense to try to use the natural order to define difference.

4.1 Monus in Naturally Ordered Monoids

▶ **Definition 3.** *Let $(M, +, 0)$ be a naturally ordered monoid. Given two elements $a, b \in M$, consider the set of all elements $z \in M$ s.t. $a + z \succeq b$. If this set has a minimal element c, then we define:*

$$b \dot{-} a \overset{def}{=} \min\{z \mid a + z \succeq b\} \tag{9}$$

Amer [3] called $(M, +, 0)$ a *Commutative Monoid with Monus*, or CMM, if it is naturally ordered and $b \dot{-} a$ exists for all $a, b \in M$. The monoid of natural numbers $(\mathbb{N}, +, 0)$ is a CMM, and its monus operation given by (9) is the same as monus in equation (3). We give two examples of classes of CMMs.

▶ **Definition 4.** *Call a naturally ordered monoid $(M, +, 0)$* complete and distributive *if $\preceq$ forms a complete, distributive lattice, and $+$ distributes over with $\bigwedge$, in other words, $x + \bigwedge\{z \mid z \in A\} = \bigwedge\{x + z \mid z \in A\}$, for any set $A \subseteq M$.*

Every complete, distributive monoid is a CMM, and its monus operation is:

$$b \dot{-} a \overset{\text{def}}{=} \bigwedge\{z \mid a + z \succeq b\} \tag{10}$$

We check that (10) satisfies Definition 3, and for that we need to show that $\bigwedge\{z \mid a + z \succeq b\}$ is the minimum element of the set $\{z \mid a + z \succeq b\}$, in other words we need to show that $a + \bigwedge\{z \mid a + z \succeq b\} \succeq b$. This follows from the fact that $+$ distributes over $\bigwedge$: $a + \bigwedge\{z \mid a + z \succeq b\} = \bigwedge\{a + z \mid a + z \succeq b\}$ and the latter is obviously $\succeq b$.

One example of a complete, distributive monoid is $(\mathbb{N}, +, 0)$, where monus (10) is the same as Eq. (3). Another example is a Boolean algebra $(2^\Omega, \cup, \emptyset)$, where (10) is set difference $b \setminus a$.

Let $(M, \preceq)$ be a complete total order, meaning that $x \preceq y$ or $y \preceq x$ for all $x, y \in M$, and $\bigwedge A$ exists for all $A \subseteq M$. Then $(M, \preceq)$ is a distributive lattice[5], and $(M, \vee, \bot)$ is a CMM, where $x \vee y \overset{\text{def}}{=} \bigwedge\{z \mid x \preceq z, y \preceq z\}$, $\bot \overset{\text{def}}{=} \bigwedge M$ is the smallest element of M. The monus operation in (10) further simplifies to:

$$b \dot{-} a = \begin{cases} 0 & \text{when } b \preceq a \\ b & \text{when } b \succ a \end{cases} \tag{11}$$

[5] We prove the identity $x \vee \bigwedge\{z \mid z \in A\} = \bigwedge\{x \vee z \mid z \in A\}$ by considering two cases. If there exists $y \in A$, $x \succeq y$, then both sides are equal to x. Assuming $x \preceq z$ for all $z \in A$ we have $\bigwedge\{x \vee z \mid z \in A\} = \bigwedge\{z \mid z \in A\}$, and the identity follows immediately.

An example of such a CMM is $([0,\infty], \min, \infty)$, where monus (11) is the same as Eq. (4). [6]

We caution that not every naturally ordered monoid a CMM. For a counterexample, consider the non-distributive lattice M_3 with elements $0 < a, b, c < 1$, where $a \vee b = a \vee c = b \vee c = 1$ and $a \wedge b = a \wedge c = b \wedge c = 0$. Then $(M_3, \vee, 0)$ is naturally ordered, but it is not a CMM because the set of z's for which $a \vee z \geq 1$ is $\{b, c, 1\}$ and it has no smallest element.

4.2 Monus as an Equational Class

In 1965 [5] Bosbach proved a remarkable result, which implies that CMMs form an equational class.[7] Amer [3] presented a simplified statement of Bosbach' result, and claimed (without proof) that CMMs are precisely the equational class defined by the axioms $(A1 - A4)$ below. We will show here Amer's identities, and give a simplified proof of Bosbach's result.

Amer's identities are the following:

$$a + (b \dot{-} a) = b + (a \dot{-} b) \tag{A1}$$
$$(a \dot{-} b) \dot{-} c = a \dot{-} (b + c) \tag{A2}$$
$$a \dot{-} a = 0 \tag{A3}$$
$$0 \dot{-} a = 0 \tag{A4}$$

▶ **Theorem 5.** *[3, 5] A commutative monoid $\boldsymbol{M} = (M, +, 0)$ is a CMM iff there exists an operation $\dot{-}$ that satisfies $(A1) - (A4)$. In particular, the class of CMMs is the restriction to the signature Σ_m of an equational class of $\Sigma_m \cup \{\dot{-}\}$ algebras.*

Proof. Assume first that $\boldsymbol{M}$ is a CMM, and let $b \dot{-} a$ be given as in Definition 3. We prove that $\boldsymbol{M}$ satisfies $(A1)$ and $(A2)$, and leave it up to the reader to check $(A3 - A4)$. By assumption $\boldsymbol{M}$ is naturally ordered, with partial order $\preceq$. Identity $(A1)$ follows from these implications:

$$\begin{aligned}
a + (b \dot{-} a) &\succeq b && \text{Definition of } b \dot{-} a \\
\exists z, a + (b \dot{-} a) &= b + z && \text{Definition of } \succeq && (12)\\
a &\preceq b + z && \text{From } a \preceq a + (b \dot{-} a) \\
a \dot{-} b &\preceq z && \text{Definition of } a \dot{-} b && (13)\\
a + (b \dot{-} a) &\succeq b + (a \dot{-} b) && \text{From (12) and (13)}
\end{aligned}$$

The opposite inequality $a + (b \dot{-} a) \preceq b + (a \dot{-} b)$ is proven similarly, and this implies $(A1)$. For $(A2)$, we start by proving $(a \dot{-} b) \dot{-} c \succeq a \dot{-} (b + c)$. By definition $a \dot{-} (b + c)$ is the smallest z satisfying the condition $z + (b + c) \succeq a$, hence it suffices to prove that $(a \dot{-} b) \dot{-} c$ also satisfies this condition. This follows from:

$$((a \dot{-} b) \dot{-} c) + b + c = (((a \dot{-} b) \dot{-} c) + c) + b \succeq (a \dot{-} b) + b \succeq a$$

[6] A small variation is the monoid $(\mathbb{R} \cup \{\infty\}, \min, \infty)$. This is also a CMM, with monus defined by the same Eq. (4). In both these CMM's the natural order $\preceq$ is the reverse of the standard order, i.e. $x \preceq y$ if $x \geq y$. This probably confused the authors of [9], who claimed incorrectly in Example 4 that $(\mathbb{R} \cup \{\infty\}, \min, \infty)$ is not a CMM.

[7] Bosbach considered naturally ordered semigroups $(M, +)$ (i.e. monoids without 0, and not necessarily commutative), which he called *holoids*, and defined a *complemented holoid* (komplementäres Holoid) to be a holoid where $b \dot{-} a$ given as in Definition 3 exists for all $a, b \in M$. Then he proved that the set of complemented holoids is an equational class, defined by just four identities over the signature $\{+, \dot{-}\}$. He further proved that every complemented holoid $(M, +, \dot{-})$ is also a commutative monoid, meaning that $+$ is commutative and has an identity, namely $0 \overset{\text{def}}{=} x \dot{-} x$, which, he showed, is independent on the choice of x.

Similarly, for the opposite inequality $a \dot{-} (b + c) \succeq (a \dot{-} b) \dot{-} c$, it suffices to prove that $(a \dot{-} (b + c)) + c \succeq a \dot{-} b$, and, for that, it suffices to prove that $((a \dot{-} (b + c)) + c) + b \succeq a$. This follows immediately by writing the inequality as $(a \dot{-} (b + c)) + (b + c) \succeq a$.

We now prove the interesting part: if $(M, +, 0)$ is a commutative monoid and admits a difference operation $\dot{-}$ that satisfies $(A1 - A4)$, then M is a CMM. Recall that $a \preceq b$ is defined as: $\exists x, a + x = b$. We first establish a simple property:

$$(P1): \qquad a \preceq b \text{ iff } a \dot{-} b = 0$$

In one direction, if $a \preceq b$ then $a \dot{-} b = a \dot{-} (a + x) = (a \dot{-} a) \dot{-} x$ (by $(A2)$) $= 0 \dot{-} x = 0$ (by $(A3), (A4)$). In the other direction, we have $b = b + 0 = b + (a \dot{-} b) = a + (b \dot{-} a)$ (by $(A1)$) which implies $b \succeq a$ by definition.

$(P1)$ implies that $(M, +, 0)$ is naturally ordered. Indeed, if both $a \dot{-} b = 0$ and $b \dot{-} a = 0$ hold, then, by $(A1)$: $a = a + 0 = a + (b \dot{-} a) = b + (a \dot{-} b) = b + 0 = b$.

It remains to prove that $(M, +, 0)$ is a CMM. For this purpose we prove a second property:

$$(P2): \qquad b \preceq a + z \text{ iff } b \dot{-} a \preceq z$$

In one direction, we use $(P1)$ and $b \preceq a + z$ to derive $0 = b \dot{-} (a + z) = (b \dot{-} a) \dot{-} z$ (by $(A2)$) and we use again $(P1)$ to conclude $b \dot{-} a \preceq z$. In the other direction, we add a to both sides of $z \succeq b \dot{-} a$ and derive $a + z \succeq a + (b \dot{-} a) = b + (a \dot{-} b)$ (by $(A1)$) $\succeq b$ (by definition of $\succeq$). We prove now that, for any two elements $a, b \in M$, the operation $b \dot{-} a$ satisfies the condition in Definition 3. On one hand $a + (b \dot{-} a) = b + (a \dot{-} b) \succeq b$. On the other hand, if z also satisfies $a + z \succeq b$, then by $(P2)$ we have $b \dot{-} a \preceq z$. Thus, $b \dot{-} a$ is the smallest element z with this property, as required. ◀

4.3 Monus in Naturally Ordered Semirings

Geerts and Poggi [9] extended CMMs from monoids to semirings. They defined an *m-semiring* to be a semiring $\boldsymbol{S} = (S, +, \cdot, 0, 1)$ where the monoid $(S, +, 0)$ is a CMM; i.e. $\boldsymbol{S}$ is naturally ordered and $b \dot{-} a \overset{\text{def}}{=} \min\{z \mid a + z \succeq b\}$ exists for elements $a, b \in S$. In particular, monus satisfies identities $(A1 - A4)$. One drawback of m-semirings is that monus is defined using only the additive monoid $(S, +, 0)$, ignoring the multiplicative operator: this creates problems, as we see next.

Tannen [4] considered the use of m-semirings as annotations of relations, and asked whether the identities $(A1 - A4)$ are sufficient to capture identities of the relational algebra. In particular, they considered the following relational algebra identity:

$$(R - S) \bowtie T = R \bowtie T - S \bowtie T \tag{14}$$

In order for (14) to hold when the relations R, S, T are annotated with values from an m-semiring, the semiring must satisfy the following identity:

$$(A5) \qquad (b \dot{-} a) \cdot c = b \cdot c \dot{-} a \cdot c$$

However, $(A5)$ does not hold in general.[8] For example, it holds in $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$, and, by extension, in the m-semiring of polynomials[9] $(\mathbb{N}[X], +, \cdot, 0, 1, \dot{-})$, but it fails in the following semiring (adapted from [4]): $\boldsymbol{S} = (\{0, \frac{1}{2}, 1\}, \vee, \wedge, 0, 1)$, where monus is given by equation (11). Here $(1 \dot{-} \frac{1}{2}) \wedge 0 = 1 \wedge 0 = 0$ while $(1 \wedge \frac{1}{2}) \dot{-} (\frac{1}{2} \wedge \frac{1}{2}) = \frac{1}{2} \dot{-} \frac{1}{2} = 0$, thus $(A5)$ fails.

[8] Inequality does hold in one direction, namely $(b \dot{-} a) \cdot c \succeq b \cdot c \dot{-} a \cdot c$, because $a + (b \dot{-} a) \succeq b$ implies $a \cdot c + (b \dot{-} a) \cdot c \succeq b \cdot c$, and, by property $(P2)$, we obtain $b \dot{-} a \succeq b \cdot c \dot{-} a \cdot c$.

[9] Monus on polynomials is defined by applying it to each monomial. For example $5x \dot{-} 2x = 3x$, and $2x \dot{-} 5x = 0$, and $x \dot{-} y = x$.

An intriguing observation by Geerts and Poggi [9] is that, although $(\mathbb{N}[X], +, \cdot, 0, 1, \dot{-})$ is an m-semiring, it is not the freely generated m-semiring.[10] The reason is that monus in $(\mathbb{N}[X], +, \cdot, 0, 1, \dot{-})$ is not defined "freely". For example, consider $\boldsymbol{B} = (\{0,1\}, \vee, \wedge, 0, 1, \dot{-})$, which is an m-semiring where $x \dot{-} y = x \wedge \neg y$. Given two variables, $X = \{x, y\}$, and the function $h : \{x, y\} \to \mathbb{B}$, $h(x) = h(y) = 1$, its unique extension $\bar{h} : \mathbb{N}[X] \to \mathbb{B}$ to a semiring homomorphism fails to be a homomorphism of m-semirings, because, on one hand, $\bar{h}(y \dot{-} x) = \bar{h}(y) = 1$, while $\bar{h}(y) \dot{-} \bar{h}(x) = 1 \dot{-} 1 = 0$.

In summary, there are two pieces of bad news for defining difference using the natural order. On one hand, m-semirings do not satisfy $(A5)$ in general, which implies that some optimizations performed by a traditional query optimizer may fail when the relations are interpreted over m-semirings. On the other hand, if we restrict only to m-semirings that satisfy $(A5)$, then we no longer have a familiar freely generated semiring. Tannen [4] suggested a deeper investigation of the freely generated m-semiring satisfying $(A5)$. However, it turns out that this is only a partial solution: such semirings will ensure that the optimization (14) is sound, but may fail other optimization rules. In fact, the only way to ensure that all relational algebra identities that are valid over bag semantics remain valid over m-semirings is to require the latter to satisfy all identities satisfied by $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$. This set of identities is co-r.e. complete, and, therefore not finitely axiomatizable. We defer the proof to Appendix A.

4.4 Natural Order and Equational Classes

Bosbach's result [5] that Commutative Monoid with Monus (CMMs) form an equational class is surprising, because it is not obvious how to define a natural order using only algebraic operations and identities. Here we investigate whether such a definition is possible. More precisely, we ask: do the naturally ordered monoids form an equational class? Similarly, do the naturally ordered semirings form an equational class? Bosbach's result does not answer this question, because it only concerns a subclass of naturally ordered monoids (semirings), namely those where monus exists. We answer the general question both negatively and positively!

First, the negative answer:

▶ **Lemma 6.** *The naturally ordered monoids are not an equational class.*
The naturally ordered semirings are not an equational class.

Proof. We will use the following known fact, which is also easy to check directly: for any signature Σ and surjective homomorphism $h : \boldsymbol{A} \to \boldsymbol{B}$, if $\boldsymbol{A}$ satisfies a set of identities E, then so does $\boldsymbol{B}$. We will fix Σ to be either $\Sigma_m = \{+, 0\}$ (the signature of monoids (1)) or $\Sigma_s = \{+, \cdot, 0, 1\}$ (the signature of semirings (2)). Assume that the set of ordered monoids (or semirings) are the equational class defined by a set of Σ_m-identities E. Consider the monoid $\boldsymbol{A} \stackrel{\text{def}}{=} \mathbb{N} \times \mathbb{N}$ where $+$ is defined component-wise, $(a, b) + (c, d) \stackrel{\text{def}}{=} (a + c, b + d)$. $\boldsymbol{A}$ is naturally ordered (this is easily verified), thus, by our assumption satisfies the identities E. Consider the homomorphism $h : \boldsymbol{A} \to \boldsymbol{Z}$, $h(a, b) \stackrel{\text{def}}{=} a - b$. Since h is surjective, we conclude that $\boldsymbol{Z}$ also satisfies E, thus, by our assumption, it is naturally ordered, which is a contradiction, proving the lemma for monoids. To prove the lemma for semirings it suffices to define the multiplication operator on $\mathbb{N} \times \mathbb{N}$ as $(a, b) \cdot (c, d) \stackrel{\text{def}}{=} (ac + bd, ad + bc)$; then h is also a homomorphism of semirings. ◀

[10] This follows immediately from the fact that $\mathbb{N}[X]$ satisfies $(A5)$, while some m-semirings don't.

However, naturally ordered monoids and semirings *are* an equational class, if we are allowed to use an additional operator, denote it $\vee$. More precisely, we extend the signature of monoids to $\Sigma_m \cup \{\vee\} = \{+, 0, \vee\}$ (Σ_m was defined in Equations (1)), and extend similarly the signature of semirings to $\Sigma_s \cup \{\vee\}$. Let E_m be the following set of $(\Sigma_m \cup \{\vee\})$-identities:

- The monoid identities.
- Semi-lattice identities for $\vee$: associativity, commutativity, and idempotence. Recall that these identities define a partial order $\leq$ by $x \leq y$ if $x \vee y = y$.
- The identity: $x \vee (x + y) = x + y$.

We define similarly the set of $(\Sigma_s \cup \{\vee\})$-identities E_s by extending the semiring identities with those for $\vee$ shown above. We prove the following result, which appears to be new:

▶ **Theorem 7.** *The class of ordered monoids is equal to the class of $(\Sigma_m \cup \{\vee\}), E_m$-algebras restricted to the monoid operators Σ_m.*

The class of ordered semirings is equal to the class of $(\Sigma_s \cup \{\vee\}), E_s$-algebras restricted to the semiring operators Σ_s.

We prove only the first statement; the second is similar. In one direction, if $(M, +, 0, \vee)$ is a $(\Sigma_m \cup \{\vee\}), E_m$-algebra, then we show that $(M, +, 0)$ is naturally ordered. Let $\leq$ be the partial ordered defined by $\vee$ (thus $a \leq b$ if $a \vee b = b$) and let $\preceq$ be the natural preorder in Eq. (8). We notice that $\preceq$ implies $\leq$, because $a + z = b$ implies $a \vee b = a \vee (a + z) = a + z = b$. Therefore $\preceq$ is antisymmetric (because $\leq$ is a partial order), proving the claim.

For the opposite direction, consider a naturally ordered monoid $(M, +, 0)$, and let $\preceq$ be its natural order. By Szpilrajn's extension theorem [17], there exists a total order $\leq$ that is an extension of $\preceq$, i.e. $a \preceq b$ implies $a \leq b$, and $\leq$ is a total order (a.k.a. linear order). Define $x \vee y \overset{\text{def}}{=} \max(x, y)$. We check that $(M, +, 0, \vee)$ is a $(\Sigma_m \cup \{\vee\}), E_m$-algebra. The only non-trivial identity is $x \vee (x + y) = x + y$. This follows from the fact that $x \preceq x + y$ (by the definition of the natural order $\preceq$), which implies $x \leq x + y$, proving that $x \vee (x + y) = x + y$.

5 Difference by Construction

A third approach to defining a difference operator in a semiring $\boldsymbol{S}$ is to construct from $\boldsymbol{S}$ some semiring $\hat{\boldsymbol{S}}$ which has a difference operator. The intuition comes from the standard method of defining $\mathbb{Z}$ from $\mathbb{N}$: first define a semiring on the product $\mathbb{N} \times \mathbb{N}$ by setting $(x, x') + (y, y') \overset{\text{def}}{=} (x + x', y + y')$ and $(x, x') \cdot (y, y') \overset{\text{def}}{=} (xy + x'y', x'y + xy')$, then consider the equivalence classes $(\mathbb{N} \times \mathbb{N})/\equiv$, where $(x, x') \equiv (y, y')$ if $x + y' = x' + y$. We explore in this section what happens if we apply a similar construction to some arbitrary semiring.

5.1 A Product Construction

▶ **Definition 8.** *Fix a semiring $\boldsymbol{S}$. We define the Σ_{sm}-algebra $\hat{\boldsymbol{S}} \overset{def}{=} (S \times S, +, \cdot, (0, 0), (1, 0), -)$, where the operations are:*

$$(x, x') + (y, y') \overset{def}{=} (x + y, x' + y') \qquad (x, x') \cdot (y, y') \overset{def}{=} (xy + x'y', xy' + x'y)$$

$$(x, x') - (y, y') \overset{def}{=} (x + y', x' + y) \qquad \mathbf{0} \overset{def}{=} (0, 0) \quad \mathbf{1} \overset{def}{=} (1, 0)$$

(where xy stands for $x \cdot y$, etc.)

We prove:

▶ **Lemma 9.** *$\hat{\boldsymbol{S}}$ is a semiring that also satisfies identity (7), and (A2), (A5).*

$$\begin{tikzcd}
 & \top \overset{\text{def}}{=} (1,1) \arrow[dl, dash] \arrow[dr, dash] & \\
\mathbf{1} \overset{\text{def}}{=} (1,0) \arrow[dr, dash] & & \bar{\mathbf{1}} \overset{\text{def}}{=} (0,1) \arrow[dl, dash] \\
 & \bar{0} \overset{\text{def}}{=} (0,0) &
\end{tikzcd}$$

$\times$	$\mathbf{0}$	$\mathbf{1}$	$\bar{\mathbf{1}}$	$\top$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\mathbf{1}$	$\mathbf{0}$	$\mathbf{1}$	$\bar{\mathbf{1}}$	$\top$
$\bar{\mathbf{1}}$	$\mathbf{0}$	$\bar{\mathbf{1}}$	$\mathbf{1}$	$\top$
$\top$	$\mathbf{0}$	$\top$	$\top$	$\top$

$-$	$\mathbf{0}$	$\mathbf{1}$	$\bar{\mathbf{1}}$	$\top$
$\mathbf{0}$	$\mathbf{0}$	$\bar{\mathbf{1}}$	$\mathbf{1}$	$\top$
$\mathbf{1}$	$\mathbf{1}$	$\top$	$\mathbf{1}$	$\top$
$\bar{\mathbf{1}}$	$\bar{\mathbf{1}}$	$\bar{\mathbf{1}}$	$\top$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

Figure 1 The semiring $\hat{\mathbb{B}}$, as per Definition 8. It is naturally ordered, with a Hasse diagram shown on the left. Addition $+$ is the LUB of the order relation, while multiplication and minus are shown in the table.

Proof. The proof that $\hat{\boldsymbol{S}}$ is a semiring is immediate, and omitted. We check identity (7) and axioms $(A2), (A5)$ directly:

$$\begin{aligned}
\text{Eq (7)}: \qquad \hat{x} + (\hat{y} - \hat{z}) =& (x, x') + ((y, y') - (z, z')) = (x + y + z', x' + y' + z)\\
(\hat{x} + \hat{y}) - \hat{z} =& ((x, x') + (y, y')) - (z, z') = (x + y + z', x' + y + z)
\end{aligned}$$

$$\begin{aligned}
(A2): \qquad (\hat{x} - \hat{y}) - \hat{z} =& (((x, x') - (y, y')) - (z, z'))\\
=& (x + y' + z', x' + y + z)\\
=& ((x, x') - ((y, y') + (z, z'))) = \hat{x} - (\hat{y} + \hat{z})
\end{aligned}$$

$$\begin{aligned}
(A5): \qquad (\hat{y} - \hat{x}) \cdot \hat{z} =& (((y, y') - (x, x')) \cdot (z, z'))\\
=& ((y + x')z + (y' + x)z', (y + x')z' + (y' + x)z)\\
=& ((yz + y'z') + (x'z + xz'), (yz' + y'z) + (x'z' + xz))\\
=& (yz + y'z', yz' + y'z) - (x'z' + xz, x'z + xz')\\
=& (y, y') \cdot (z, z') - (x, x') \cdot (z, z') = \hat{y} \cdot \hat{z} - \hat{x} \cdot \hat{z}
\end{aligned}$$
◀

In general $\hat{\boldsymbol{S}}$ does not satisfy $(A1)$ and $(A4)$, but this is not a problem, for example, they don't hold in any ring either.

▶ **Example 10.** If $\mathbb{B}$ is the Boolean semiring, then $\hat{\mathbb{B}}$ consists of four elements, which we denote as $\mathbf{0}, \mathbf{1}, \bar{\mathbf{1}}, \top$, and are shown in Figure 1. The elements can be interpreted as follows: $\mathbf{0}$ = false, $\mathbf{1}$ = positive, $\bar{\mathbf{1}}$ =negative, and $\top$ = over specified (both positive and negative).

More generally, let's apply this construction to a Boolean algebra $\boldsymbol{S} = (2^\Omega, \cup, \cap, \emptyset, \Omega)$. The elements of $\hat{\boldsymbol{S}}$ are pairs of sets (A, B), and can be best viewed as functions $\nu : \Omega \to \hat{\mathbb{B}}$ mapping the elements in the four sets $\Omega \setminus (A \cup B), A \setminus B, B \setminus A, A \cap B$ to $\mathbf{0}, \mathbf{1}, \bar{\mathbf{1}}$, and $\top$ respectively. For example the pair $(\{a, b, d, e, f\}, \{c, d, f\})$ in $\hat{\boldsymbol{S}}$ can be denote more friendly as $\{a, b, \bar{c}, d\bar{d}, e, f\bar{f}\}$, meaning that the elements a, b, e are positive (inserted), c is negative (removed), while d and f were over specified (both inserted and removed).

However, so far we only used the first step from the standard construction of the integers from the natural numbers: taking the cross product. In many cases we need the second step as well: taking the quotient w.r.t. some congruence relation. Generalizing the $\mathbb{N}$-to-$\mathbb{Z}$ construction, our first attempt is to define $\equiv$ as the smallest a congruence relation on $\bar{\boldsymbol{S}}$ satisfying:

$$x + y' = x' + y \implies (x, x') \equiv (y, y') \tag{15}$$

The problem with this definition is that the quotient semiring may become trivial: for example, both $\hat{\mathbb{B}}/\equiv$ and $\widehat{\mathsf{Trop}}/\equiv$ are trivial. To see this, notice that both semirings have an absorptive element $\top$ satisfying $\top + x = \top$, and therefore $(x, x') \equiv (\top, \top)$ for all (x, x'). Thus, we will not consider definition (15), and instead will define a congruence on $\hat{\boldsymbol{S}}$ by using a semiring ideal.

5.2 Ideals in Semirings

Recall that an *ideal* in a ring $\boldsymbol{R}$ is a subset $I \subseteq R$ s.t. $x, y \in I$ implies $x + y \in I$, and $x \in I, u \in R$ implies $u \cdot x \in I$. The equivalence relation $x \equiv_I y$ defined by $x - y \in I$ is a congruence, and the set $R/\equiv_I$ is called the *quotient ring*. We generalize these concepts from rings to semirings.

▶ **Definition 11.** *An* ideal *in a semiring $\boldsymbol{S}$ is a set $I \subseteq S$ satisfying $u, v \in I \Rightarrow u + v \in I$, and $u \in I, x \in S \Rightarrow u \cdot x \in I$. Given an ideal I, we define two congruence relations:*

$$x \equiv_I y \quad \text{if} \quad \exists u, v \in I, x + u = y + v \tag{16}$$

$$x \cong_I y \quad \text{if} \quad \{(a,b) \mid a, b \in S, ax + b \in I\} = \{(a,b) \mid a, b \in S, ay + b \in I\} \tag{17}$$

It can be checked immediately that both $\equiv_I$ and $\cong_I$ are congruence relations.[11] Therefore both quotients $\boldsymbol{S}/\equiv_I$ and $\boldsymbol{S}/\cong_I$ are semirings, and the canonical mappings $\boldsymbol{S} \to \boldsymbol{S}/\equiv_I$ and $\boldsymbol{S} \to \boldsymbol{S}/\cong_I$ are semiring homomorphisms. Both the definition of an ideal, and of the congruence relation $\equiv_I$ appear in the literature [14–16], and are extensively covered in [10]. The definition of the congruence relation $\cong_I$ appears to be novel.

In a ring, both $\equiv_I, \cong_I$ are equal, and are the same as the standard congruence relation defined by the ideal I; moreover, the congruence class $0/\cong_I$ is precisely the ideal I. For a semiring $\boldsymbol{S}$ and any set A s.t. $0 \in A \subseteq S$, we define the *closure* as $cl(A) \stackrel{\text{def}}{=} \{x \mid \exists a \in A, a + x \in A\}$. We prove:

▶ **Lemma 12.** *In any semiring $\boldsymbol{S}$, (1) $I \subseteq cl(I) = 0/\equiv_I$ and (2) any congruence class $x/\cong_I$ is either a subset of I, or disjoint from I. In particular, $0/\cong_I \subseteq I$.*

Proof. The first statement is immediate, and is well known in the literature. For the second statement, we prove that if $x/\cong_I$ contains some element $u \in I$, then it is a subset of I. Assume $x \cong_I u$ and $u \in I$. Then $x \in I$ follows by setting $a = 1, b = 0$ in (17): then $au + b = u \in I$, and therefore $ax + b = x \in I$. ◀

Henriksen [14] called an ideal I a *k-ideal* if $I = cl(I)$; see also [2, 15, 16]. For example, in the semiring of natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$, the set $I = \{6k + 8\ell \mid k, \ell \in \mathbb{N}\}$ is an ideal $I = \{0, 6, 8, 12, 14, 16, \ldots\}$ which is not a k-ideal: $cl(I)$ is the set of all even numbers, and $cl(I) - I = \{2, 4, 10\}$. We prove:

▶ **Lemma 13.** *The following statements are equivalent: (1) I is a k-ideal, (2) $0/\equiv_I = I$, (3) $0/\cong_I = I$.*

Proof. The equivalence of (1) and (2) is immediate, and was well known in the literature. To prove (1) $\Rightarrow$ (3) we show that for all $u \in I$, $u \cong_I 0$. Let $a, b \in S$ be such that $au + b \in I$; then $au \in I$ and, since $I = cl(I)$, we have $b \in I$, implying $a0 + b \in I$. Conversely, if $a0 + b \in I$, then $b \in I$ and we have $au + b \in I$ because $au \in I$. To prove (3) $\Rightarrow$ (1), assume $u \in I$ and $u + b = 1 \cdot u + b \in I$. Since $u \cong_I 0$ we also have $1 \cdot 0 + b \in I$, proving $b \in I$. ◀

[11] The rationale behind having the parameter a in (17), as opposed to fixing $a = 1$, is to ensure that $\cong_I$ is a congruence w.r.t. multiplication.

We return now to our product semiring $\hat{\boldsymbol{S}}$ in Definition 8, and define the following ideal in $\hat{\boldsymbol{S}}$.

$$\Delta \overset{\text{def}}{=} \{(x,x) \mid x \in S\} \tag{18}$$

It can be checked immediately that Δ is an ideal in $\hat{\boldsymbol{S}}$, which we call the *diagonal ideal.* In general, Δ is not a k-ideal, and, therefore, the congruences $\equiv_\Delta$ and $\cong_\Delta$ are distinct. One can check that each of them is also a congruence w.r.t. to the difference operator. We examine now the quotients $\boldsymbol{S}/\equiv_\Delta$ and $\boldsymbol{S}/\cong_\Delta$, starting with $\equiv_\Delta$.

▶ **Lemma 14.** *For any semiring $\boldsymbol{S}$, the Σ_{sm}-algebra $\hat{\boldsymbol{S}}/\equiv_\Delta$ is the ring freely generated by $\boldsymbol{S}$.*

Proof. By Lemma 9 $\hat{\boldsymbol{S}}$ satisfies the semiring identities and identity (7), and therefore so does $\hat{\boldsymbol{S}}/\cong_\Delta$. It remains to prove it satisfies identity (6): $\hat{x} - \hat{x} = (x,x') - (x,x') = (x+x', x+x') \in \Delta$, which implies $(\hat{x}-\hat{x})/\equiv_\Delta = \hat{0}/\equiv_\Delta$, as required. To prove that it is the freely generated ring, we check the diagram (5) from Theorem 1: given a ring $\boldsymbol{R}$ and a semiring homomorphism $h : \boldsymbol{S} \to \boldsymbol{R}$, first extend it to a Σ_{sm}-homomorphism $\hat{h} : \hat{S} \to \boldsymbol{R}$ by $\hat{h}(x,x') \overset{\text{def}}{=} h(x) - h(x')$, then observe that $(x,x') \equiv_\Delta (y,y')$ implies that there exists $(u,u) \in \Delta$ such that $(x+u, x'+u) = (y+u, y'+u)$, which implies $\hat{h}(x+u, x'+u) = h(x+u) - h(x'+u) = h(x) - h(x') = \hat{h}(y+u, y'+u) = h(y) - h(y')$, in other words $\hat{h}(x,x') = \hat{h}(y,y')$. Therefore, we can uniquely extend $\hat{h} : \hat{\boldsymbol{S}} \to \boldsymbol{R}$ to $\hat{\boldsymbol{S}}/\equiv_\Delta \to \boldsymbol{R}$. This completes the proof of the lemma. ◀

The lemma gives us a constructive way to obtain the ring freely generated by the semiring $\boldsymbol{S}$, but, as we saw in Lemma 2, the freely generated ring can sometimes be trivial. This justifies exploring the second alternative for our construction: $\hat{\boldsymbol{S}}/\cong_\Delta$. We prove:

▶ **Lemma 15.** *For any semiring $\boldsymbol{S}$, $\hat{\boldsymbol{S}}/\cong_\Delta$ is a semiring that satisfies the identities* (7)*, and* $(A2), (A5)$*. Moreover: (1) the mapping $\eta : \boldsymbol{S} \to \hat{\boldsymbol{S}}/\cong_\Delta$, $\eta(x) \overset{def}{=} (x,0)/\cong_\Delta$ is an injective homomorphism, and (2) the mapping $x \mapsto (0,x)/\cong_\Delta$ is injective (but not a homomorphism).*

Proof. By Lemma 9 $\hat{\boldsymbol{S}}$ satisfies the identities (7), and $(A2), (A5)$, therefore so does $\hat{\boldsymbol{S}}/\cong_\Delta$. We prove (1). It is straightforward to check that η is a homomorphism, we prove that it is injective. Assume $(x,0) \cong_\Delta (y,0)$ and set $\hat{a} = (1,0)$, $\hat{b} = (0,x)$ in (17). Then $\hat{a} \cdot (x,0) + \hat{b} = (x,x) \in \Delta$, and therefore we must have $\hat{a} \cdot (y,0) + \hat{b} = (y,x) \in \Delta$, which implies $x = y$ as required. The proof of (2) is similar and omitted. ◀

The lemma proves that $\hat{\boldsymbol{S}}/\cong_\Delta$ contains two copies of $\boldsymbol{S}$:

- a copy $\{(x,0)/\cong_I \mid x \in S\}$ of elements that we call *positive elements*, and
- a copy $\{(0,x)/\cong_I \mid x \in S\}$ of elements that we call *negative elements.*

Our last result proves that, in the case of the semirings $\mathbb{B}$ and Trop, then quotients $\hat{\mathbb{B}}/\cong_\Delta$ and $\widehat{\mathsf{Trop}}/\cong_\Delta$ consists precisely of the positive elements, negative elements, and *over determined* elements $\{(x,x)/\cong_I \mid x \in S\}$. We prove this separately for Boolean algebras and for Trop.

▶ **Lemma 16.** *$\hat{\mathbb{B}}/\cong_I$ is isomorphic to $\hat{\mathbb{B}}$ (shown in Fig. 1). As a consequence, if $\boldsymbol{S}$ is a Boolean algebra, as in Example 10, then $\hat{\boldsymbol{S}}/\cong_\Delta$ is isomorphic to $\hat{\boldsymbol{S}}$.*

Proof. It suffices to check that no two elements in $\hat{\mathbb{B}}$ are congruent, by checking six inequalities of the form $\hat{x} \cong_\Delta \hat{y}$. In each case we will show that there exists $\hat{b}$ such that $\hat{x} + \hat{b} \in \Delta$ and $\hat{y} + \hat{b} \notin \Delta$ (in other words, $\hat{a} = \mathbf{1}$ in all cases):

- $\mathbf{0} \ncong_\Delta \mathbf{1}$ and $\top \ncong_\Delta \mathbf{1}$: choose $\hat{b} = \bar{\mathbf{1}}$.
- $\mathbf{0} \ncong_\Delta \bar{\mathbf{1}}$ and $\top \ncong_\Delta \bar{\mathbf{1}}$: choose $\hat{b} = \mathbf{1}$.
- $\mathbf{1} \ncong_\Delta \bar{\mathbf{1}}$: choose $\hat{b} = \bar{\mathbf{1}}$.
- $\top \ncong_\Delta \mathbf{0}$: choose $\hat{b} = \mathbf{1}$. ◀

We prove next the same result for Trop, which we state in a slightly more general form. Recall that a *diod* is a semiring $\boldsymbol{S}$ where addition is idempotent. It can be shown that a diod is naturally ordered, and addition is the LUB, in other words $\boldsymbol{S} = (S, \vee, \cdot, 0, 1)$. Call a diod *strict* if its natural order $\preceq$ is total, and multiplication is cancelative, meaning that $a \cdot x = a \cdot y$ implies $x = y$ when $a \neq 0$; the semiring Trop is strict. We prove:

▶ **Lemma 17.** *Let $\boldsymbol{S}$ be a strict diod. Then $\hat{\boldsymbol{S}}/\cong_\Delta$ consists of the following congruence classes:*

- *Zero,* $(0,0)/\cong_\Delta = \{(0,0)\}$.
- *The* positive *elements* $(x,0)/\cong_\Delta = \{(x,z) \mid x \succ z\}$*, for* $x \in S$*,* $x \neq 0$.
- *The* negative *elements* $(0,x)/\cong_\Delta = \{(z,x) \mid z \prec x\}$*, for* $x \in S$*,* $x \neq 0$
- *The* over determined *elements* $(x,x)/\cong_\Delta = \{(x,x)\}$*, for* $x \in S$*,* $x \neq 0$.

Proof. Let $\sim$ be the following equivalence relation on $\hat{\boldsymbol{S}}$:

$$(x,y) \sim (u,v) \quad \text{if } \left((y \prec x = u \succ v) \text{ or } (x \prec y = v \succ u) \text{ or } (x = y = u = v)\right)$$

To prove the lemma we have to show that $\cong_\Delta = \sim$.

We start by showing $\cong_\Delta \subseteq \sim$, and for that we show that $(x,y) \not\sim (u,v)$ implies $(x,y) \not\cong_\Delta (u,v)$. Assume $(x,y) \not\sim (u,v)$. There are three cases. **Case 1**: $x \succ y$ and $u = v$. Setting $\hat{a} \overset{\text{def}}{=} (1,0)$ and $\hat{b} \overset{\text{def}}{=} (0,0)$ we have

$$\hat{a} \cdot (x,y) \bigvee \hat{b} = (x,y) \notin \Delta$$
$$\hat{a} \cdot (u,u) \bigvee \hat{b} = (u,u) \in \Delta$$

proving $(x,y) \cong_\Delta (u,u)$. **Case 2**: $x \succ y$ and $u \prec v$. Then we set $\hat{a} \overset{\text{def}}{=} (1,0)$, $\hat{b} \overset{\text{def}}{=} (v,0)$ and we have:

$$\hat{a} \cdot (x,y) \bigvee \hat{b} = (x,y) \bigvee (v,0) = (x \vee v, y) \notin \Delta \text{ because } x \vee v \succeq x \succ y$$
$$\hat{a} \cdot (u,v) \bigvee \hat{b} = (u,v) \bigvee (v,0) = (u \vee v, v) = (v,v) \in \Delta$$

Case 3: $x \prec y$ and $u = v$ is similar to case 1 and omitted.

Next, we prove that $(x,y) \sim (u,v)$ implies $(x,y) \cong_\Delta (u,v)$. For that it suffices to assume that $y \prec x = u \succ v$: the second case $x \prec y = v \succ u$ is symmetric, and the third case $x = y = u = v$ is trivial. Thus, it suffices to prove:

$$\text{If } y \prec x \succ v \text{ then } (x,y) \cong_\Delta (x,v) \tag{19}$$

We apply the definition of $\cong_\Delta$ in Eq. (17) to (x,y) and (x,v). It suffices to prove that $\hat{a} \cdot (x,y) \bigvee \hat{b} \in \Delta$, implies $\hat{a} \cdot (x,v) \bigvee \hat{b} \in \Delta$, for any two elements $\hat{a} = (a,a')$ and $\hat{b} = (b,b')$ in $\hat{S}$; the other direction of the implication is proven similarly and we will omit it. The semiring operations in $\hat{\boldsymbol{S}}$ were defined in Def. 8, and the condition $\hat{a} \cdot (x,y) \bigvee \hat{b} \in \Delta$ is equivalent to:

$$ax \vee a'y \vee b = ay \vee a'x \vee b' \tag{20}$$

which can be further rewritten two:

$$\begin{aligned}(ax \vee b) \vee a'y &= (a'x \vee b') \vee ay \\ A \vee a'y &= A' \vee ay\end{aligned} \tag{21}$$

where $A \overset{\text{def}}{=} ax \vee b \succeq ay$ and $A' \overset{\text{def}}{=} a'x \vee b' \succeq a'y$.

We claim that condition (21) and the fact that the semiring $\boldsymbol{S}$ is strict implies:

$$A = A' \tag{22}$$

The claim completes the proof because, equality (21) continues to hold if we replace y with v, because $A \succeq av$, $A' = A \succeq a'v$ and therefore $A \vee a'v = A$ and $A' \vee ay = A \vee ay = A$. Thus, it remains to prove that (21) and the fact that $\boldsymbol{S}$ is strict implies (22).

From (21) we derive:

$$A \vee a'y = A' \vee ay = (A \vee a'y) \vee (A' \vee ay) = A \vee A' \tag{23}$$

If $a = a' = 0$ then we immediately obtain $A = A'$.

Assume w.l.o.g. that $a \neq 0$. Since $\boldsymbol{S}$ is strict and $x \succ y$, we derive $ax \succ ay$ and therefore $A \succ ay$. It means that the four equal quantities in (23) are $\succ ay$. Since $\preceq$ is a total order, the least upper bound $A' \vee ay$ is either A' or $A' \vee ay = ay$: the latter impossible (because we proved that $A' \vee ay \succ ay$), therefore all four quantities in (23) are equal to A'.

In particular it holds that $A \vee a'y = A'$. We now consider two cases. When $a' = 0$, then we immediately derive $A = A'$. When $a' \neq 0$ then $A' \succ a'y$: since the least upper bound $A \vee a'y$ is either A or $a'y$, and it cannot be $a'y$, it that it is equal to A. Since all terms in (23) are equal to A', we conclude $A = A'$, as required. ◀

6 Discussion

We have examined three alternative ways to add difference to a semiring: by specifying the desired identities, by using the natural order, or by construction. The construction-based approach appears to be novel: we have only investigated a couple of options for the construction and proved only a few properties, leaving many open questions. For example, one open question is whether the class of Σ_{sm}-algebras of the form $\hat{\boldsymbol{S}}/\cong_\Delta$ is an equational class.

However, our investigation is far from complete. We mention here only one example that deserves further exploration. Grädel and Tannen [11] and later Dannert, Grädel, Naaf and Tannen [8] gave an interpretation to negative information by considering dual-indeterminate polynomials, $\mathbb{N}[X, \bar{X}]$. Such a polynomial has two kinds of variables (also called *provenance tokens*): positive variables x and negative variables $\bar{x}$. For example $3x + 4y\bar{z}^2 + \bar{x}\bar{z}$. They further assumed $x\bar{x} = 0$ for every variable x, which is equivalent to taking the quotient w.r.t. the ideal I generated by the monomials of the form $x\bar{x}$. One can naturally define a difference operator in this semiring as $f - g \overset{\text{def}}{=} f + \bar{g}$, where $\bar{g}$ is obtained by converting each variable from positive to negative and vice versa. Thus, $\mathbb{N}[X, \bar{X}]$ (or, more precisely, $\mathbb{N}[X, \bar{X}]/\equiv_I$) becomes an Σ_{sm}-algebra. What are the identities (in addition to the semiring identities) satisfied by this algebra? Is there any connection to the $\hat{\boldsymbol{S}}/\cong_\Delta$ construction that we explored in Sec. 5? We leave these questions for future work.

References

1 Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, Dan Suciu, and Yisu Remy Wang. Convergence of datalog over (pre-) semirings. In Leonid Libkin and Pablo Barceló, editors, *PODS '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 105–117. ACM, 2022. doi:10.1145/3517804.3524140.

2 P. J. Allen. A fundamental theorem of homomorphisms for semirings, 1969. URL: https://api.semanticscholar.org/CorpusID:8723880.

3 K Amer. Equationally complete classes of commutative monoids with monus. *Algebra Universalis*, 18:129–131, 1984. doi:10.1007/BF01182254.

4 Yael Amsterdamer, Daniel Deutch, and Val Tannen. On the limitations of provenance for queries with difference. In Peter Buneman and Juliana Freire, editors, *3rd Workshop on the Theory and Practice of Provenance, TaPP'11, Heraklion, Crete, Greece, June 20-21, 2011*. USENIX Association, 2011. URL: https://www.usenix.org/conference/tapp11/limitations-provenance-queries-difference.

5 Bruno Bosbach. Komplementäre Halbgruppen. Ein Beitrag zur instruktiven Idealtheorie kommutativer Halbgruppen. *Math. Annalen*, 161:279–295, 1965.

6 Stanley Burris and Hanamantagouda P. Sankappanavar. *A course in universal algebra*, volume 78 of *Graduate texts in mathematics*. Springer, 1981.

7 Shumo Chu, Brendan Murphy, Jared Roesch, Alvin Cheung, and Dan Suciu. Axiomatic foundations and algorithms for deciding semantic equivalences of SQL queries. *Proc. VLDB Endow.*, 11(11):1482–1495, 2018. doi:10.14778/3236187.3236200.

8 Katrin M. Dannert, Erich Grädel, Matthias Naaf, and Val Tannen. Semiring provenance for fixed-point logic. In Christel Baier and Jean Goubault-Larrecq, editors, *29th EACSL Annual Conference on Computer Science Logic, CSL 2021, January 25-28, 2021, Ljubljana, Slovenia (Virtual Conference)*, volume 183 of *LIPIcs*, pages 17:1–17:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.CSL.2021.17.

9 Floris Geerts and Antonella Poggi. On database query languages for k-relations. *J. Appl. Log.*, 8(2):173–185, 2010. doi:10.1016/j.jal.2009.09.001.

10 Jonathan S Golan. *Semirings and their Applications*. Springer Science & Business Media, 2013.

11 Erich Grädel and Val Tannen. Semiring provenance for first-order model checking. *CoRR*, abs/1712.01980, 2017. arXiv:1712.01980.

12 Todd J. Green, Zachary G. Ives, and Val Tannen. Reconcilable differences. In Ronald Fagin, editor, *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, volume 361 of *ACM International Conference Proceeding Series*, pages 212–224. ACM, 2009. doi:10.1145/1514894.1514920.

13 Todd J. Green, Gregory Karvounarakis, and Val Tannen. Provenance semirings. In Leonid Libkin, editor, *Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China*, pages 31–40. ACM, 2007. doi:10.1145/1265530.1265535.

14 M. Henriksen. Ideals in semirings with commutative addition. *Notices of the American Mathematical Society*, 6:321, 1958.

15 D. R. Latorre. A note on quotient semirings. *Proceedings of the American Mathematical Society*, 24(3):463–465, 1970. URL: http://www.jstor.org/stable/2037388.

16 M. Sen and Mahim Adhikari. On k-ideals of semirings. *International Journal of Mathematics and Mathematical Sciences*, 15, January 1992. doi:10.1155/S0161171292000437.

17 Edward Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae*, 16(1):386–389, 1930. URL: http://eudml.org/doc/212499.

A Monus and Query Optimization

A natural question emerging from [4] is to find a set of m-semiring identities E that is sound for query optimization. We show here that such a set is undecidable, and, in fact, it is co-r.e.-complete. More precisely, we seek a set of identities E such that, if two relational algebra (RA) queries are equivalent under standard semantics, then they remain equivalent over K-relations, when we use a semiring that satisfies E. In that case we say that the set E is *sound*. A trivial sound set is $E = \{0 = 1\}$, because it only holds in the trivial semiring, where all queries becomes equivalent. To avoid such degenerate solutions, we ask for a *minimal* sound set of identities E. We describe here this minimal set, by considering two flavors of soundness, depending on what semantics we adopt for RA expressions: set, or bag semantics. For example, if we use set semantics, then an optimizer could replace $R \cup R$ with R, and we need to add idempotence ($x + x = x$) to E to ensure soundness, but for bag semantics we don't need idempotence. In this section we prove the following. The minimal sound set of identities for bag semantics is $E_{\Sigma_{sm}}(\mathbb{N})$, i.e. the set of all identities satisfied by $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$: this set is co-r.e.-complete, and, thus, undecidable, and not finitely axiomatizable. Thus, adding just ($A5$), or any finite set of identities to the m-semirings identities is insufficient to ensure soundness. We also prove that the minimal sound set of identities for set semantics is $E_{\Sigma_{sm}}(\mathbb{B})$, the set of identities satisfied by $(\mathbb{B}, \vee, \wedge, 0, 1, \dot{-})$, which are the identities of Boolean algebras: there are well known finite axiomatizations for this set.

To state and prove this result formally we need a brief review of relational algebra and its interpretation over K-relations, based on [13].

Relational algebra[12], RA, consists of the six operators $\bowtie, \sigma, \Pi, \cup, -, \rho$. ($\rho$ is "renaming".) If we drop $-$, then it is called the *positive relational algebra*, and denoted RA^+. Recall the signatures Σ_s and Σ_{sm} from Eq. (2). Let $\boldsymbol{S}$ be any Σ_s-algebra (not necessarily a semiring), and define an $\boldsymbol{S}$-relation of arity k to be a function $R : \mathsf{Dom}^k \to \boldsymbol{S}$ of finite support (i.e. $\{t \mid R(t) \neq 0\}$ is finite). When $R(t) = u \in \boldsymbol{S}$, then we say that a tuple t is *annotated* with the element u. Tannen [13] associated to each operator in RA^+ an operation on $\boldsymbol{S}$-relations, in a natural way. For example union $R_1 \cup R_2$ returns the $\boldsymbol{S}$-relation $(R_1 \cup R_2)(t) \overset{\text{def}}{=} R_1(t) + R_2(t)$, natural join returns $(R_1 \bowtie R_2)(t) \overset{\text{def}}{=} R_1(\Pi_{\text{attrs}(R_1)}(t_1)) \cdot R_2(\Pi_{\text{attrs}(R_2)}(t_2))$, etc;[13] we refer the reader to Definition 3.2. in [13]. If $\boldsymbol{S}$ is an Σ_{sm} algebra, then we extend this definition from RA^+ to RA, by defining $(R_1 - R_2)(t) \overset{\text{def}}{=} R_1(t) - R_2(t)$.

Let Q_1, Q_2 be two RA-expressions. We write $Q_1 \equiv_{\boldsymbol{S}} Q_2$ if these two expressions return the same output for any input $\boldsymbol{S}$-relations. For example, if $+$ is commutative in $\boldsymbol{S}$ and $Q_1 = R \cup R'$, $Q_2 = R' \cup R$, then $Q_1 \equiv_{\boldsymbol{S}} Q_2$. Since $\mathbb{N}$-relations are bags, the equivalence $Q_1 \equiv_{\mathbb{N}} Q_2$ holds iff Q_1, Q_2 are equivalent RA-expressions under bag semantics. Similarly, $Q_1 \equiv_{\mathbb{B}} Q_2$ iff Q_1, Q_2 are equivalent under set semantics, for example $R \cup R \equiv_{\mathbb{B}} R$.

[12] The term *algebra* in RA is used with some abuse, since it is not a Σ-algebra, in the sense of Sec. 3. This is because the operators can only be applied to arguments with the right schemas, for example $R \cup S$ is defined only if the relations R, S have the same arity.

[13] Notice that the result of an operation may be an $\boldsymbol{S}$-relation with infinite support; this was apparently overlooked in [13]. However, this does not affect either the results in [13], nor those in this section, because, when $\boldsymbol{S}$ is a semiring, then all operations return $\boldsymbol{S}$-relations with finite support, assuming the inputs also have finite support.

▶ **Definition 18.** *A set of Σ_{sm}-identities E is* sound *for RA under bag semantics if, for any Σ_{sm}, E-algebra $\boldsymbol{S}$ and any two RA queries Q_1, Q_2, if $Q_1 \equiv_{\mathbb{N}} Q_2$ then $Q_1 \equiv_{\boldsymbol{S}} Q_2$. Similarly, E is sound for RA under set semantics if, for any Σ_{sm}, E-algebra $\boldsymbol{S}$ and any two RA queries Q_1, Q_2, if $Q_1 \equiv_{\mathbb{B}} Q_2$ then $Q_1 \equiv_{\boldsymbol{S}} Q_2$.*

Similarly, a set of Σ_s-identities is sound for RA$^+$ under bag (set) semantics if the condition above holds when Q_1, Q_2 are restricted to RA$^+$.

Our goal is to find a minimal set E that is sound for bag (set) semantics.

If $\boldsymbol{S}$ is any Σ_{sm} algebra then we denote by $E_{\text{RA}}(\boldsymbol{S})$ the set of identities $Q_1 \equiv_{\boldsymbol{S}} Q_2$, where Q_1, Q_2 are RA queries, and denote by $E_{\text{RA}}(\mathcal{C}) \stackrel{\text{def}}{=} \bigcap_{\boldsymbol{S}\in\mathcal{C}} E_{\text{RA}}(\boldsymbol{S})$ where $\mathcal{C}$ is a class of Σ_s-algebras. We define similarly $E_{\text{RA}^+}(\boldsymbol{S}), E_{\text{RA}^+}(\mathcal{C})$ by restricting to Σ_s algebras and RA$^+$ queries. Tannen answered the soundness question for RA$^+$ under bag semantics in Proposition 3.4 of [13]:

▶ **Theorem 19** (Implicit in [13]). *(a) $E_{RA^+}(\mathbb{N}) \subseteq E_{RA^+}(\boldsymbol{S})$ iff $\boldsymbol{S}$ is a semiring. (b) If $\mathcal{C}$ is an equational class of Σ_s-algebras, then $E_{RA^+}(\mathcal{C}) = E_{RA^+}(\mathbb{N})$ iff $\mathcal{C}$ is the class of semirings.*

Part (a) proves that if E are the identities of semirings, then E is sound for RA$^+$ and bag semantics. Part (b) proves that E is the smallest sound set of identities. We will prove below a more general result that extends Theorem 19 from RA$^+$ to RA; the same proof can be used to prove Theorem 19.

Before we can extend the theorem, we need a brief review of Galois connections. Given two sets U, V, a Galois connection is a pair of functions $F : 2^U \to 2^V$, $G : 2^V \to 2^U$ such that (a) F, G are anti-monotone, and (b) the following condition holds:

$$Y \subseteq F(X) \text{ iff } X \subseteq G(Y) \tag{24}$$

In any Galois connection the following hold: $F(G(F(X))) = F(X)$ and $G(F(G(Y))) = G(Y)$.

Consider now a signature Σ, and an infinite set of variables X, and recall that a Σ-identity is a pair $e = (e_1, e_2)$ where $e_1, e_2 \in T_\Sigma(X)$. For any Σ-algebra $\boldsymbol{A}$, denote by $E_\Sigma(\boldsymbol{A})$ the set of identities that hold on $\boldsymbol{A}$. For example, $E_{\Sigma_{sm}}(\mathbb{N})$ contains all identities satisfied by $\mathbb{N}$, which includes the semiring identities, the CMM identities $(A1) - (A4)$, the identity $(A5)$, and many more. Furthermore, for an identity e denote by $\mathcal{C}_\Sigma(e)$ the class of Σ-algebras that satisfy e. Then the following two mappings form a Galois connection:

$$E_\Sigma(\mathcal{C}) \stackrel{\text{def}}{=} \bigcap_{\boldsymbol{A}\in\mathcal{C}} E_\Sigma(\boldsymbol{A}) \qquad\qquad \mathcal{C}_\Sigma(E) \stackrel{\text{def}}{=} \bigcap_{e\in E} \mathcal{C}_\Sigma(e) \tag{25}$$

The generalization of Theorem 19 to RA is the following:

▶ **Theorem 20.** *(a) $E_{RA}(\mathbb{N}) \subseteq E_{RA}(\boldsymbol{S})$ iff $\boldsymbol{S}$ is a $E_{\Sigma_{sm}}(\mathbb{N})$-algebra. (b) If $\mathcal{C}$ is any equational class of Σ_{sm}-algebras, then $E_{RA}(\mathcal{C}) = E_{RA}(\mathbb{N})$ iff $\mathcal{C}$ is the class defined by the identities $E_{\Sigma_{sm}}(\mathbb{N})$.*

Thus, in order to ensure soundness, the semiring $\boldsymbol{S}$ must satisfy all identities satisfied by the m-semiring $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$, which we denoted $E_{\Sigma_{sm}}(\mathbb{N})$. Before we prove the theorem, we show that this set is co-r.e. complete.

▶ **Theorem 21.** *$E_{\Sigma_{sm}}(\mathbb{N})$ is co-r.e. complete.*

Proof. Membership in co-r.e. is immediate: to check that an identity $e_1 = e_2$ is false in $\mathbb{N}$, it suffices to iterate over all possible assignments to the variables of e_1, e_2 and stop when one such assignment makes $e_1 \neq e_2$. To prove completeness, it suffices to prove that membership is undecidable, and for that we will use Matiyasevich theorem on the undecidability of Hilbert's tenth problem. It implies that the following problem is undecidable: *given a multivariate polynomial* $F \in \mathbb{Z}[X]$ *with variables* $x_1, x_2, \ldots$ *decide if there exists values* $x_1, x_2, \ldots \in \mathbb{N}$ *s.t.* $F(x_1, x_2, \ldots) = 0$. It follows immediately that the following problem is undecidable: given two polynomials $F, G \in \mathbb{N}[X]$, decide if the following holds:

$$\exists x_1, x_2, \ldots \in \mathbb{N} : \quad F(x_1, x_2, \ldots) = G(x_1, x_2, \ldots)$$

Its negation (which is also undecidable) is the statement:

$$\forall x_1, x_2, \ldots \in \mathbb{N} : \quad F(x_1, x_2, \ldots) \neq G(x_1, x_2, \ldots) \tag{26}$$

which we abbreviate by $F \neq G$. We use the following equivalences in $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$:

$$F \neq G \text{ iff } (F \dot{-} G) + (G \dot{-} F) > 0 \text{ iff } 1 \dot{-} ((F \dot{-} G) + (G \dot{-} F)) = 0$$

Since (26) is undecidable, checking identities of the form $1 \dot{-} ((F \dot{-} G) + (G \dot{-} F)) = 0$ in the m-semiring $(\mathbb{N}, +, \cdot, 0, 1, \dot{-})$ is also undecidable. Thus, $E_{\Sigma_{sm}}(\mathbb{N})$ is undecidable, therefore co-r.e. complete. ◀

To summarize, the minimal set of identities that ensures that an m-semiring is sound for all RA-identities is the set $E_{\Sigma_{sm}}(\mathbb{N})$, which is infinite, co-r.e.-complete, and, thus, it is not finitely generated. Obviously, the set $E_{\Sigma_{sm}}(\mathbb{N})$ is not practical. A possible workaround could be to find a non-minimal sound set, which is still useful for practical purposes: we leave this for future work.

We briefly discuss what happens when we interpret RA using set semantics instead of bag semantics, or consider both semantics. Theorem 20 continues to hold if we replace $E_{\text{RA}}(\mathbb{N})$ and $E_{\Sigma_{sm}}(\mathbb{N})$ with $E_{\text{RA}}(\mathbb{B})$ and $E_{\Sigma_{sm}}(\mathbb{B})$, thus the necessary and sufficient identities in this case are $E_{\Sigma_{sm}}(\mathbb{B})$. These are precisely the identities of Boolean algebras, which are generated by a finite set, and membership is decidable. This does *not* imply that $E_{\text{RA}}(\mathbb{B})$ is decidable: in fact $E_{\text{RA}}(\mathbb{B})$ consists of all pairs of RA-expressions that are equivalent under set semantics, and is undecidable by Trakhtenbrot's theorem.

Theorems 20 also specializes to RA$^+$, and we derive the following version of Theorem 19: the minimal set of Σ_s-identities E that is sound for RA$^+$ under bag semantics is $E_{\Sigma_s}(\mathbb{N})$, which is equivalent to the semiring axioms,[14], hence we recover Theorem 19. Similarly, we obtain that the minimal set of Σ_s-identities that are sound for RA$^+$ under set semantics is $E_{\Sigma_s}(\mathbb{B})$, which is equivalent to the set of identities of bounded, distributive lattices.[15] Finally, we briefly discuss what happens if we extend RA to support mixed set/bag semantics, by adding an operator δ to RA which eliminates duplicates. This requires us to add a new operation to the semiring $\boldsymbol{S}$, lets call it also δ, which satisfies $\delta(0) = 0$ and $\delta(x) = 1$ for all $x \neq 0$. Unfortunately, the δ-semirings do not form an equational class, because the product of two δ-semirings, $\boldsymbol{S}_1 \times \boldsymbol{S}_2$, is not a δ-semiring: for $(x, 0) \in \boldsymbol{S}_1 \times \boldsymbol{S}_2$, we have $\delta(x, 0) = (\delta_1(x), \delta_2(0)) = (1, 0)$, which is neither $(0, 0)$, nor $(1, 1)$. An operation of this kind was considered in [7] under the name *squash*, and defined using a conditional axiom.

[14] To see this, consider any two expressions $e_1, e_2 \in T_{\Sigma_s}(X)$ that are equivalent in $(\mathbb{N}, +, \cdot, 0, 1)$. Using the semiring identities only we can write e_1, e_2 in a canonical form, as a sum of monomials, i.e. $e_1, e_2 \in \mathbb{N}[X]$. Since they are equivalent in $(\mathbb{N}, +, \cdot, 0, 1)$, they must be identical polynomials. Thus, the equivalence $e_1 = e_2$ follows using only semiring axioms.

[15] The proof is similar to the above. Any two expressions e_1, e_2 equivalent in $\mathbb{B}$ can be transformed into DNF expressions using only the identities of distributive lattices, and their DNF expressions must be isomorphic.

In the rest of this section we prove Theorem 20. The proof follows from three lemmas. If $f : S_1 \to S_2$ is any function, then, for any k-ary $\boldsymbol{S}_1$-relation R we will denote by $f \circ R$ the $\boldsymbol{S}_2$-relation defined by $(f \circ R)(t) \overset{\text{def}}{=} f(R(t))$, for all $t \in \mathsf{Dom}^k$. If $\bar{R} = (R_1, R_2, \ldots)$ is a tuple of relations, then we write $f \circ \bar{R}$ for $(f \circ R_1, f \circ R_2, \ldots)$.

▶ **Lemma 22.** *Let $f : \boldsymbol{S}_1 \to \boldsymbol{S}_2$ be a homomorphism between Σ_{sm}-algebra, and Q be an RA query. Let $\bar{R}$ be a tuple of $\boldsymbol{S}_1$-relations. Then $Q(f \circ \bar{R}) = f \circ Q(\bar{R})$.*

Proof. The proof follows immediately by induction on the structure of Q. We illustrate here only for the case when Q is the union of two sub-queries; all other cases are similar. Assume $Q = Q_1 \cup Q_2$ and let t be a tuple in the output. Then:

$$\begin{aligned}
(Q(f\circ\bar{R}))(t) = \left(Q_1(f\circ\bar{R}) \cup Q_2(f\circ\bar{R})\right)(t) \quad &= \left(Q_1(f\circ\bar{R})\right)(t) + \left(Q_2(f\circ\bar{R})\right)(t)\\
= \left(f\circ Q_1(\bar{R})\right)(t) + \left(f\circ Q_2(\bar{R})\right)(t) \quad &= f\left(Q_1(\bar{R})(t)\right) + f\left(Q_2(\bar{R})(t)\right)\\
= f\left(Q_1(\bar{R})(t) + Q_2(\bar{R})(t)\right) \quad &= f\left(\left(Q_1(\bar{R}) \cup Q_2(\bar{R})\right)(t)\right) = \left(f\circ Q(\bar{R})\right)(t)
\end{aligned}$$

◀

▶ **Lemma 23.** *Let $\boldsymbol{S}_1, \boldsymbol{S}_2$ be two Σ_{sm}-algebras. Then $E_{RA}(\boldsymbol{S}_1) \subseteq E_{RA}(\boldsymbol{S}_2)$ iff $E_{\Sigma_{sm}}(\boldsymbol{S}_1) \subseteq E_{\Sigma_{sm}}(\boldsymbol{S}_2)$. The same statement holds for Σ_s-algebras and RA^+.*

In other words, in order to compare the RA-identities (RA$^+$-identities) satisfied by $\boldsymbol{S}_1$-relations with those satisfied by $\boldsymbol{S}_2$-relations, it suffices to compare the algebraic identities satisfied by $\boldsymbol{S}_1$ with those satisfied by $\boldsymbol{S}_2$.

Proof. We start with the $\Leftarrow$ direction, and assume $E_{\Sigma_{sm}}(\boldsymbol{S}_1) \subseteq E_{\Sigma_{sm}}(\boldsymbol{S}_2)$. Let $(Q_1, Q_2) \in E_{\text{RA}}(\boldsymbol{S}_1)$, in other words $Q_1(\bar{R}) = Q_2(\bar{R})$ for any input $\boldsymbol{S}_1$-relation instance $\bar{R}$. Consider some input $\boldsymbol{S}_2$-instance $\bar{R}'$: we need to prove that $Q_1(\bar{R}') = Q_2(\bar{R}')$. Let X be a set of variables, s.t. $|X| = |S_2|$, let $\eta : X \to T_{\Sigma_{sm}}(X)$ be the canonical injection, $h : X \to S_2$ be a bijection, $\bar{h} : T_{\Sigma_{sm}}(X) \to \boldsymbol{S}_2$ its extension to a homomorphism, and $\bar{R}'' \overset{\text{def}}{=} h^{-1} \circ \bar{R}'$. Thus, if a tuple in $\bar{R}''$ is annotated with variable $x \in X$, then the same tuple is annotated in $\bar{R}'$ with $h(x) \in S_2$: formally, $h \circ \bar{R}'' = \bar{h} \circ \eta \circ \bar{R}'' = \bar{R}'$. Then $Q_i(\bar{R}') = Q_i(\bar{h} \circ \eta \circ \bar{R}'') = \bar{h} \circ Q_i(\eta \circ \bar{R}'')$ (by Lemma 22) for $i = 1, 2$. (We cannot apply Q_i to $\bar{R}''$ because its annotations are variables in X; instead we apply Q_i to $\eta \circ \bar{R}''$, whose annotations are the same variables, but viewed in $T_{\Sigma_{sm}}(X)$, where the Σ_{sm}-operations are defined.) Let t be some output tuple, and define:

$$e_i \overset{\text{def}}{=} (Q_i(\eta \circ \bar{R}''))(t), \quad i = 1, 2$$

Thus, e_1, e_2 are expressions in $T_{\Sigma_{sm}}(X)$ that annotate the tuple t in the outputs of Q_1, Q_2 on $\bar{R}''$ respectively. We claim that $(e_1, e_2) \in E_{\Sigma_{sm}}(\boldsymbol{S}_1)$ (i.e. they form an identity that holds in $\boldsymbol{S}_1$). For that, we need to show that for any function $g : X \to S_1$, the equality $\bar{g}(e_1) = \bar{g}(e_2)$ holds. To prove that we use the fact that Q_1, Q_2 return the same output on the $\boldsymbol{S}_1$ relation $\bar{g} \circ \eta \circ \bar{R}''$:

$$\begin{aligned}
\left(Q_1(\bar{g}\circ\eta\circ\bar{R}'')\right)(t) &= \left(Q_2(\bar{g}\circ\eta\circ\bar{R}'')\right)(t)\\
\bar{g}\left(Q_1(\eta\circ\bar{R}'')(t)\right) &= \bar{g}\left(Q_2(\eta\circ\bar{R}'')(t)\right)\\
\bar{g}(e_1) &= \bar{g}(e_2)
\end{aligned}$$

Since g was arbitrary, we conclude that $(e_1, e_2) \in E_{\Sigma_{sm}}(\boldsymbol{S}_1)$, and therefore $(e_1, e_2) \in E_{\Sigma_{sm}}(\boldsymbol{S}_2)$, which implies $\bar{h}(e_1) = \bar{h}(e_2)$, proving that $Q_1(\bar{R}')(t) = Q_2(\bar{R}'')(t)$.

We briefly sketch the $\Rightarrow$ direction of the proof, for Σ_{sm} and RA. Given $(e_1, e_2) \in E_{\Sigma_{sm}}(\boldsymbol{S}_1)$, we convert both e_1 and e_2 into RA-expressions over unary relations. For example, if e_1 is $(x_1^2 \cdot x_2 \dot{-} x_3) + x_1$ then Q_1 is $(R_1 \bowtie R_1 \bowtie R_2 - R_3) \cup R_1$, where R_1, R_2, R_3 are unary relations with the same attribute name. It follows immediately that $(Q_1, Q_2) \in E_{\mathrm{RA}}(\boldsymbol{S}_1)$, hence it is also in $E_{\mathrm{RA}}(\boldsymbol{S}_2)$, and this implies that the identity $e_1 = e_2$ also holds in $\boldsymbol{S}_2$. ◀

The last lemma is:

▶ **Lemma 24.** *Fix a signature Σ. (a) For any two Σ-algebras $\boldsymbol{A}, \boldsymbol{B}$, the following holds: $E_\Sigma(\boldsymbol{A}) \subseteq E_\Sigma(\boldsymbol{B})$ iff $\boldsymbol{B} \in \mathcal{C}_\Sigma(E_\Sigma(\boldsymbol{A}))$. (b) If $\mathcal{C}$ is an equational class of Σ-algebras, then $E_\Sigma(\mathcal{C}) = E_\Sigma(\boldsymbol{A})$ iff $\mathcal{C} = \mathcal{C}_\Sigma(E_\Sigma(\boldsymbol{A}))$.*

Proof. Part (a) is by the definition of the Galois connection (24). We prove now part (b). For one direction of (b), assume $\mathcal{C} = \mathcal{C}_\Sigma(E_\Sigma(\boldsymbol{A}))$. Then $E_\Sigma(\mathcal{C}) = E_\Sigma(\mathcal{C}_\Sigma(E_\Sigma(\boldsymbol{A}))) = E_\Sigma(\boldsymbol{A})$. For the other direction, assume $\mathcal{C}$ is an equational class and $E_\Sigma(\mathcal{C}) = E_\Sigma(\boldsymbol{A})$. Then $\mathcal{C}_\Sigma(E_\Sigma(\mathcal{C})) = \mathcal{C}_\Sigma(E_\Sigma(\boldsymbol{A}))$, and the claim follows from the fact that $\mathcal{C}_\Sigma(E_\Sigma(\mathcal{C})) = \mathcal{C}$ because $\mathcal{C}$ is an equational class, i.e. $\mathcal{C} = \mathcal{C}_\Sigma(E)$ for some E. ◀

Lemmas 23 and 24 immediately imply Theorem 20.

An Intensional Expressiveness Gap of Comprehension Syntax

Limsoon Wong
School of Computing, National University of Singapore, Singapore

Abstract

Comprehension syntax is widely adopted in modern programming languages as a means for manipulating collection types. This paper proves that all subquadratic algorithms which are expressible in comprehension syntax, do not compute low-selectivity joins. As database systems support these joins efficiently, this confirms an intensional expressiveness gap between comprehension syntax and relational database systems. The proof of this intensional expressiveness gap relies on a "limited-mixing" lemma which states that subquadratic algorithms expressible using comprehension syntax have limited ability for mixing atomic objects in their inputs.

2012 ACM Subject Classification Information systems → Relational database query languages

Keywords and phrases Comprehension syntax, intensional expressive power, limited-mixing lemma

Digital Object Identifier 10.4230/OASIcs.Tannen.2024.11

Funding *Limsoon Wong*: This work was supported in part by a Singapore Ministry of Education tier-2 grant, MOE-T2EP20221-0005; and in part by a Kwan Im Thong Hood Cho Temple Chair Professorship.

Acknowledgements During Val Tannen's visit to Singapore in 2019, we developed the concept of synchrony iterators [17]. Synchrony iterators enables efficient algorithms for synchronized iterations on multiple collections to be expressed elegantly using comprehension syntax. Notably, they simplify the creation of efficient algorithms for low-selectivity non-equijoins. During our discussions, I was struck by an apparent gap: the difference between the efficiency of database queries in comprehension syntax and their real-world implementation in relational database systems. I am grateful for Val's friendship and mentorship over the past three and a half decades. I am honoured and glad to contribute this chapter, which bridges the above gap in our understanding of comprehension syntax, as a tribute to Val in this festschrift volume.

1 Overview

Query languages based on comprehension syntax are able to express all relational queries supported by typical database systems [4, 22]. Moreover, queries written in comprehension syntax are appealingly simple [3]. So, comprehension syntax has become widely regarded as a means for embedding collection-type querying capabilities into programming languages. However, join queries expressed in comprehension syntax in these programming languages are generally compiled into nested loops. This implies such queries typically have quadratic or even higher time complexity when they are expressed by means of comprehension syntax.

In contrast, in relational database systems, even in the absence of indices, when joins have low selectivity, these joins often have $O(n \log n)$ time complexity based on (sort-)merge-join algorithms [2]. And when the input relations are pre-sorted on their join attributes in a low-selectivity join, a merge-join can even be realised with linear time complexity by skipping the sorting steps.

Therefore, there is a potential intensional expressiveness gap between algorithms that can be realised by comprehension syntax and those used in database systems, such as algorithms for low-selectivity joins. Consequently, despite the syntactic naturalness of comprehension syntax, one might say it fails as a genuine naturally embedded query language. Nonetheless, this gap has not been formally proven.

The main objective of this paper is to prove that this intensional expressiveness gap indeed exists. The proof goes via a "limited-mixing" lemma on $\mathcal{NRC}_1(\leq)$. On ordered data types, $\mathcal{NRC}_1(\leq)$ is equivalent to the flat relational algebra or first-order logic [22]. More pertinently, there is a simple translation between comprehension syntax and $\mathcal{NRC}_1(\leq)$, and this translation preserves time complexity. This makes $\mathcal{NRC}_1(\leq)$ a suitable ambient query language for investigating the potential intensional expressiveness gap between comprehension syntax and typical database systems.

The limited-mixing lemma states that all $\mathcal{NRC}_1(\leq)$ queries of subquadratic time complexity are only able to mix atoms in their input relations in very limited ways. So, these subquadratic-complexity queries cannot be low-selectivity joins. This limited-mixing lemma is non-query specific and is applicable even when ordered data types are present. It thus considerably enriches the available theoretical tools for studying intensional expressive power, as these tools are often query specific and are inapplicable in the presence of ordered data types. It is also a useful intensional counterpart to Gaifman's locality property [8]. Gaifman's locality is very useful for analyzing extensional expressiveness of first-order query languages on unordered data types, but is inapplicable to ordered data types.

This chapter is organized as follows. Section 2 presents $\mathcal{NRC}_1$, its operational semantics, rewrite rules and the induced normal forms. Section 3 states and proves the limited-mixing lemma. Section 4 leverages the limited-mixing lemma to prove the main result that all implementations of *zip*, which is a prototypical linear-time low-selectivity join, in $\mathcal{NRC}_1$ have at least quadratic time complexity. This confirms the intensional expressive power gap between comprehension syntax and relational database systems. Finally, Section 5 provides discussion on the intensional expressiveness gap and how the gap could be addressed.

2 Nested Relational Calculus

The restriction of the nested relational calculus $\mathcal{NRC}$ from Buneman et al. [4] and Wong [22] to flat relations is used as the ambient language here. $\mathcal{NRC}$ is equivalent to the usual nested relational algebra [4, 22]. Its restriction to flat relations, denoted here as $\mathcal{NRC}_1$, is equivalent to flat relational algebra and first-order logic [22]. This ambient language, and its operational semantics and rewrite rules, are described below.

2.1 Types and expressions

The types and expressions of $\mathcal{NRC}$ are given in Figure 1. The type superscripts in the figure are omitted when there is no confusion. For simplicity, all variable names are assumed to be distinct. For convenience, all data types are endowed with an order; this query language is denoted as $\mathcal{NRC}(\leq)$.

The semantics of a type is just a set of objects built up by nesting sets and records of base-type objects. Base types are denoted by b (representing atomic values in a database). An object of type $s_1 \times \cdots \times s_n$ is a tuple (i.e., a record) whose ith component is an object of type s_i, for $1 \leq i \leq n$. An object of type $\{s\}$ is a finite set whose elements are objects of type s; an object of type $\{s\}$ is called a set or a "relation." Moreover, if $s = b \times \cdots \times b$, then an object of type $\{s\}$ (or s) is called a "flat relation." However, if s contains some set brackets, then an object of type $\{s\}$ is called a "nested relation."

Types in $\mathcal{NRC}$

$$s ::= b \mid s_1 \times \cdots \times s_n \mid \{s\}$$

where b is a base type.

Expressions in $\mathcal{NRC}$

$$\frac{}{C^s : s} \qquad \frac{}{x^s : s} \qquad \frac{e_1 : s_1 \quad \ldots \quad e_n : s_n}{(e_1, \ldots, e_n) : s_1 \times \cdots \times s_n} \qquad \frac{e : s_1 \times \cdots \times s_n}{e.\pi_i : s_i} 1 \leq i \leq n$$

$$\frac{}{\{\}^s : \{s\}} \qquad \frac{e : s}{\{e\} : \{s\}} \qquad \frac{e_1 : \{s\} \quad e_2 : \{s\}}{e_1 \cup e_2 : \{s\}} \qquad \frac{e_1 : \{s\} \quad e_2 : \{t\}}{\bigcup\{e_1 \mid x^t \in e_2\} : \{s\}}$$

$$\frac{}{\mathit{true} : \mathbb{B}} \qquad \frac{}{\mathit{false} : \mathbb{B}} \qquad \frac{e_1 : \mathbb{B} \quad e_2 : s \quad e_3 : s}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 : s}$$

$$\frac{e_1 : s \quad e_2 : s}{e_1 < e_2 : \mathbb{B}} \qquad \frac{e_1 : s \quad e_2 : s}{e_1 = e_2 : \mathbb{B}} \qquad \frac{e : \{s\}}{e\ \mathit{isempty} : \mathbb{B}}$$

Figure 1 $\mathcal{NRC}$.

The expression constructs are defined as follows. The expression C denotes objects, including constants of base types b; the syntax for C will be given in the next subsection. The expression $(e_1, \ldots, e_n)$ forms a tuple whose ith component is the object denoted by e_i, for $1 \leq i \leq n$. The expression $e.\pi_i$ extracts the ith component of the tuple e. The expressions $\{\}$, $\{e\}$, and $e_1 \cup e_2$ have their conventional meaning as set operations. The expression $\bigcup\{e_1 \mid x \in e_2\}$ forms the set obtained by first applying the function $f(x) = e_1$ to each object in the set e_2 and then taking their union; that is, $\bigcup\{e_1 \mid x \in e_2\} = f(C_1) \cup \ldots \cup f(C_n)$, where $f(x) = e_1$ and $\{C_1, \ldots, C_n\}$ is the set denoted by e_2.

Besides the object types and their expression constructs above, $\mathcal{NRC}$ also has the Boolean type $\mathbb{B}$ as a base type, and the expression constructs *true*, *false*, and *if* e_1 *then* e_2 *else* e_3, which have their conventional meaning as Boolean values and conditional expression. Lastly, the expression $e_1 < e_2$ provides a linear ordering on objects of the same type; the expression $e_1 = e_2$ checks whether the objects denoted by e_1 and e_2 are the same; and the expression e *isempty* checks whether the set denoted by e is empty.

The emptiness test e *isempty*, the equality test $e_1 = e_2$, and the ordering test $e_1 < e_2$ are provided for every type s solely for convenience. They are actually defined in terms of the tests on base types b. In particular, the linear ordering on any arbitrarily deeply nested combinations of record and set types can be lifted – in a manner definable by $\mathcal{NRC}$ – from the linear ordering on each base type b as follows [13]: for tuple types $s_1 \times \cdots \times s_n$, it is defined pointwise lexicographically; and for set types $\{s\}$, it is defined a la Wechler [21] based on the Hoare ordering (viz. $X \leq Y$ iff for all $x \in X - Y$, there is $y \in Y - X$, such that $x \leq y$).

The notation $x \in e_2$ in the $\bigcup\{e_1 \mid x \in e_2\}$ construct is an abstraction that introduces the variable x whose scope is the expression e_1. That is, it is part of the syntax and is not a membership test. This construct is the sole means in $\mathcal{NRC}$ for iterating over a set.

If a variable appearing in an expression e is not introduced by a subexpression of the form $\bigcup\{e_1 \mid x \in e_2\}$ in e, it is called a free variable of e. When it is necessary to explicitly indicate the free variables of an expression, we write $e(x_1, ..., x_2)$ or $e(\vec{x})$. An expression $e(\vec{x})$, with free variables $\vec{x}$ can be regarded as a function $f(\vec{x}) = e(\vec{x})$. When it is desirable to distinguish the free variables local to a subexpression $e(\vec{x}, \vec{X})$ of an expression $e'(\vec{X})$, uppercase is used for the free variables of the entire expression while lowercase is used for other free variables of the subexpression. Also, an expression e having no free variable is called a closed expression.

When objects $\vec{C}$ have the same types as the free variables $\vec{x}$ of an expression $e(\vec{x})$, the expression obtained by replacing each variable x_i in $\vec{x}$ in $e(\vec{x})$ by the corresponding C_i in $\vec{C}$ is denoted as $e[\vec{C}/\vec{x}]$. The result of applying $e(\vec{x})$ as a function to $\vec{C}$ is denoted by $e(\vec{C})$. To make notations lighter, $e(\vec{C})$ is sometimes also used to denote the expression $e[\vec{C}/\vec{x}]$; however, this usage is generally eschewed in proofs.

A "pattern-matching" construct $\bigcup\{e_1 \mid (x_1, \ldots, x_n) \in e_2\}$ is used for convenience. It is a syntactic sugar for $\bigcup\{e_1[x.\pi_1/x_1, \ldots, x.\pi_n/x_n] \mid x \in e_2\}$. There is also an easy mechanical translation [3, 22] between the syntax of $\mathcal{NRC}$ and comprehension syntax of the form $\{e \mid \delta_1, \ldots, \delta_n\}$ where each δ_i either has the form $\vec{x}_i \in e_i$ or the form e_i. The translation is as follows:

- $\{e \mid \vec{x}_1 \in e_1, \Delta\} =_{df} \bigcup\{\{e \mid \Delta\} \mid \vec{x}_1 \in e_1\}$;
- $\{e \mid e_1, \Delta\} =_{df}$ *if* e_1 *then* $\{e \mid \Delta\}$ *else* $\{\}$; and
- $\{e \mid \} =_{df} \{e\}$.

Comprehension syntax is used here to write examples, but the reader should understand these examples as syntactic sugars of the actual $\mathcal{NRC}$ expressions.

▶ **Example 1.** All relational queries [5] are expressible in $\mathcal{NRC}$.

- $\Pi_i\ X =_{df} \{x.\pi_i \mid x \in X\}$ is the relational projection;
- $\sigma_d\ X =_{df} \{x \mid x \in X,\ d(x)\}$ is the relational selection;
- $X \bowtie Y =_{df} \{(x, y) \mid (u, x) \in X,\ (v, y) \in Y,\ u = v\}$ is the relational join;
- $X \cap Y =_{df} \{x \mid x \in X,\ \textit{not}\ \{y \mid y \in Y,\ y = x\}\ \textit{isempty}\}$ is the relational intersection.
- $X - Y =_{df} \{x \mid x \in X,\ \{y \mid y \in Y,\ y = x\}\ \textit{isempty}\}$ is the relational difference; and
- $X \div Y =_{df} \{x \mid (x, y) \in X,\ Y \subseteq \{y' | (x', y') \in X,\ x' = x\}\}$, where $Y \subseteq Y' =_{df} Y - Y'\ \textit{isempty}$, is the relational division.

▶ **Example 2.** $\mathcal{NRC}$ can also express nested relational operations [20].

- *unnest* $R =_{df} \{(x, y)|\ (X, y) \in R, x \in X\}$ unnests the nested relation R; and
- *nest* $R =_{df} \{(\{x \mid (x, y) \in R, y = v\}, v) \mid (u, v) \in R\}$ creates a nested version of a relation R, which groups values in the first column of R by values in the second column of R.

Let $\mathcal{NRC}_1$ denote the fragment of $\mathcal{NRC}$ where expressions are restricted to flat relation types. That is, in $\mathcal{NRC}_1$, every (sub)expression $e(x_1, ..., x_n) : s$ where $x_i : s_i$ for $1 \leq i \leq n$, the types s, s_1, ..., s_n are all flat relations. It is known that $\mathcal{NRC}$ enjoys the conservative extension property [22]; thus, $\mathcal{NRC}(\leq)$ and $\mathcal{NRC}_1(\leq)$ express the same functions on flat relations, and are equivalent to flat relational algebra or first-order logic with ordering $FO(\leq)$.

▶ **Proposition 3.** *$\mathcal{NRC}(\leq)$, $\mathcal{NRC}_1(\leq)$, and $FO(\leq)$ have the same extensional expressive power on flat relations.*

An expression $e(\vec{x})$ in $\mathcal{NRC}$ can always be turned into an expression $e'(\vec{y}, \vec{x})$ such that no constants or objects appear in it. This can be obtained by introducing fresh free variables $\vec{y}$ and replacing each object C_i in $e(\vec{x})$ by the variable y_i; then $e'[\vec{C}/\vec{y}](\vec{x}) = e(\vec{x})$. So, for simplicity, and without loss of generality, only constant-free expressions are considered when results are stated and proved in this paper.

$$\frac{}{C \Downarrow C}$$

$$\frac{e_1 \Downarrow C_1 \quad \ldots \quad e_n \Downarrow C_n}{(e_1, \ldots, e_n) \Downarrow (C_1, \ldots, C_n)} \qquad \frac{e \Downarrow (C_1, \ldots, C_n)}{e.\pi_i \Downarrow C_i}\, 1 \leq i \leq n$$

$$\frac{}{\{\} \Downarrow \{\}} \qquad \frac{e \Downarrow C}{\{e\} \Downarrow \{C\}} \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 \cup e_2 \Downarrow C_1 \oplus C_2}$$

$$\frac{\begin{array}{c} e_2 \Downarrow \{C_1, \ldots, C_n\} \\ e_1[C_1/x] \Downarrow C'_1 \quad \cdots \quad e_1[C_n/x] \Downarrow C'_n \end{array}}{\bigcup\{e_1 \mid x \in e_2\} \Downarrow C'_1 \oplus \cdots \oplus C'_n}$$

$$\frac{}{\mathit{true} \Downarrow \mathit{true}} \qquad \frac{}{\mathit{false} \Downarrow \mathit{false}}$$

$$\frac{e_1 \Downarrow \mathit{true} \quad e_2 \Downarrow C}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \Downarrow C} \qquad \frac{e_1 \Downarrow \mathit{false} \quad e_3 \Downarrow C}{\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3 \Downarrow C}$$

$$\frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 < e_2 \Downarrow \mathit{true}}\, C_1 < C_2 \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 < e_2 \Downarrow \mathit{false}}\, C_1 \not< C_2$$

$$\frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow \mathit{true}}\, C_1 = C_2 \qquad \frac{e_1 \Downarrow C_1 \quad e_2 \Downarrow C_2}{e_1 = e_2 \Downarrow \mathit{false}}\, C_1 \neq C_2$$

$$\frac{e \Downarrow C}{e\ \mathit{isempty} \Downarrow \mathit{true}}\, C = \{\} \qquad \frac{e \Downarrow C}{e\ \mathit{isempty} \Downarrow \mathit{false}}\, C \neq \{\}$$

Figure 2 A call-by-value operational semantics of $\mathcal{NRC}$.

2.2 Operational semantics

In order to discuss intensional expressive power, i.e. what algorithms are expressible, it is necessary to know how an expression of $\mathcal{NRC}$ is executed. This is specified in Figure 2 as a call-by-value operational semantics. A call-by-value operational semantics is widely adopted in programming languages and has also been used for several variations of $\mathcal{NRC}$ in earlier works [18, 19, 24] on intensional expressive power.

In Figure 2, the notation $e \Downarrow C$ means the closed expression e is evaluated to produce the object C. The unique evaluation tree of e is denoted using the notation $e \Downarrow$. The "step" complexity $\mathit{step}(e \Downarrow)$ of an evaluation is defined as the time complexity of the largest node in the evaluation tree – viz., $\mathit{step}(e \Downarrow) = \max\{\mathit{time}(e' \Downarrow C') \mid$ the node $e' \Downarrow C'$ occurs in the evaluation tree of $e \Downarrow$. The time complexity $\mathit{time}(e' \Downarrow C')$ of a node is the number of branches that the node has. E.g., in Figure 2, $\mathit{time}(\bigcup\{e_1 \mid x \in e_2\} \Downarrow C'_1 \oplus \cdots \oplus C'_n) = n + 1$. On the other hand, the time complexity $\mathit{time}(e \Downarrow)$ of an evaluation is the sum of the time complexity of all the nodes in the tree.

The syntax for objects C is as follows. A constant c of a base type b is an object of type b. A tuple $(C_1, \ldots, C_n)$ is an object of type $s_1 \times \cdots \times s_n$ if each C_i is an object of type s_i. An "enumeration list", elist for short, $\{C_1, \ldots, C_n\}$ is an object of type $\{s\}$ if each C_i is an object of type s. An elist $\{C_1, .., C_n\}$ can be thought of as a particular way of enumerating the elements of the set that it represents, viz. C_1 followed by C_2, followed by C_3, and so on. There are as many distinct elists that represent the same set as there are distinct ways to enumerate elements of that set, corresponding to different ordering and multiplicity of appearances of its elements in the enumeration.

The notations $C = C'$ and $C == C'$ are used to refer to two notions of equality involving elists. The notation $C = C'$ means C are C' are the same objects when all the elists contained in them (and objects therein) are interpreted as sets: thus, $c = c'$ iff c and c' are the same constant of a base type; $(C_1, ..., C_n) = (C'_1, ..., C'_n)$ iff $C_i = C'_i$ for $1 \leq i \leq n$; and $\{C_1, ..., C_n\} = \{C'_1, ..., C'_m\}$ iff for each $1 \leq i \leq n$, there is $1 \leq j \leq m$ such that $C_i = C'_j$, and for each $1 \leq j \leq m$, there is $1 \leq i \leq n$ such that $C_i = C'_j$. The notation $C == C'$ means C and C' are the same objects when all the elists contained in them (and objects therein) are interpreted as lists: thus, $c == c'$ iff c and c' are the same constant of a base type; $(C_1, ..., C_n) == (C'_1, ..., C'_n)$ iff $C_i == Ci'$ for $1 \leq i \leq n$; and $\{C_1, ..., C_n\} == \{C'_1, ..., C'_m\}$ iff $n = m$, and $C_i == C'_i$ for $1 \leq i \leq n$.

In Figure 2, a constructor $C \oplus C'$ is used to produce the concatenation of two elists in constant time; i.e. given $C == \{C_1, ..., C_n\}$ and $C' == \{C'_1, ..., C'_m\}$, $C \oplus C' == \{C_1, ..., C_n, C'_1, ..., C'_m\}$. Also, $\oplus$ is always used in a right-associative manner; e.g., $C \oplus C' \oplus C''$ means $C \oplus (C' \oplus C'')$. Note that while it is not a common practice to use a constant-time concatenation constructor to represent lists, it has been used in e.g. the influential Kleisli Query System [23] which is based on $\mathcal{NRC}$.

Linear orderings $<$ are available on all base types and are lifted to all types, as defined earlier. With this, the subset of objects in "canonical form" can be defined as follows. A constant c of any base type b is canonical. A tuple $(C_1, ..., C_n)$ is canonical if each C_i is canonical. An elist $\{C_1, ..., C_n\}$ is canonical if for every $1 \leq i, j \leq n$, it is the case that C_i is canonical, C_j is canonical, and $C_i < C_j$ iff $i < j$; a canonical elist is thus duplicate-free and is sorted according to $<$. The notation $\mathit{canonize}(C)$ denotes the unique canonical form of the object C. Clearly, for $C == \{C_1, ..., C_n\}$ representing a flat relation, $\mathit{canonize}(C)$ can be produced in $O(n \log(n))$ time.

The call-by-value operational semantics in Figure 2 does not perform canonization. This is because canonization is not needed to guarantee the soundness of an evaluation in $\mathcal{NRC}(\leq)$.

▶ **Proposition 4** (Soundness). *Suppose $e(\vec{x}) : s$ is an expression in $\mathcal{NRC}$, $\vec{C}$ are objects having the same types as $\vec{x}$, and $e[\vec{C}/\vec{x}] \Downarrow C'$. Then $e[\vec{C}/\vec{x}] = C'$.*

The size of an object C can be defined in any reasonable way. One way is defining $\mathit{size}(C)$ as the number of symbols used to write C out. Another way, when C is an elist, is defining $\mathit{size}(C)$ as $|C|$, the length of the elist. Both notions of size can be generalized to $\mathit{size}(\vec{C}) = \sum_i \mathit{size}(C_i)$. The latter notion of input size is used by default. Then the time complexity of an expression $e(\vec{x})$ can be defined in the usual way based on input size; i.e. the time complexity of $e(\vec{x})$ is a function $g : \mathbb{N} \to \mathbb{N}$ where $g(n)$ equals the maximum of $\mathit{time}(e[\vec{C}/\vec{x}] \Downarrow C')$ over all inputs $\vec{C}$ of size at most n. Then, the time complexity is said to be constant if g is $\Theta(1)$, linear if g is $\Theta(n)$, quadratic if g is $\Theta(n^2)$, and polynomial if g is $\Theta(n^k)$ for some natural number k. The following is easily shown in a manner similar to [4, Theorem 4.4].

▶ **Proposition 5** (Polynomiality). *Let $e(\vec{x}) : s$ be an expression in $\mathcal{NRC}(\leq)$. Then there is a number k such that the time complexity of $e(\vec{x})$ is $\Theta(n^k)$ where n denotes input size. In particular, if the time complexity of $e(\vec{x})$ is sub-quadratic, then it must be either linear or constant time; and if it is sub-linear, then it must be constant time. Furthermore, these properties are retained when $\mathcal{NRC}$ is augmented by any additional functions that have polynomial time complexity.*

$$
\begin{array}{rcl}
\bigcup\{e \mid x \in \{\}\} & \mapsto & \{\} \\
\bigcup\{e_1 \mid x \in \{e_2\}\} & \mapsto & e_1[e_2/x] \\
\bigcup\{e \mid x \in (e_1 \cup e_2)\} & \mapsto & \bigcup\{e \mid x \in e_1\} \cup \bigcup\{e \mid x \in e_2\} \\
\bigcup\{e_1 \mid x \in \bigcup\{e_2 \mid y \in e_3\}\} & \mapsto & \bigcup\{\bigcup\{e_1 \mid x \in e_2\} \mid y \in e_3\} \\
\bigcup\{e \mid x \in (\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3)\} & \mapsto & \mathit{if}\ e_1\ \mathit{then}\ \bigcup\{e \mid x \in e_2\}\ \mathit{else}\ \bigcup\{e \mid x \in e_3\} \\
(e_1, \ldots, e_2).\pi_i & \mapsto & e_i \\
(\mathit{if}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3).\pi_i & \mapsto & \mathit{if}\ e_1\ \mathit{then}\ e_2.\pi_i\ \mathit{else}\ e_3.\pi_i \\
\mathit{if}\ \mathit{true}\ \mathit{then}\ e_2\ \mathit{else}\ e_3 & \mapsto & e_2 \\
\mathit{if}\ \mathit{false}\ \mathit{then}\ e_2\ \mathit{else}\ e_3 & \mapsto & e_3
\end{array}
$$

Figure 3 A system of rewrite rules for $\mathcal{NRC}$.

2.3 Rewrite rules

Figure 3 shows a system of rewrite rules for simplifying $\mathcal{NRC}$ expressions. These rules have been used in many previous works on $\mathcal{NRC}$ [22, 14, 15, 24]. These rules are easily shown to be sound, and do not increase step complexity, and are strongly normalizing [22].

Although this system of rewrite rules does not increase step complexity, it can increase time complexity. E.g., rewriting $\bigcup\{\bigcup\{\{(x,z)\} \mid z \in Z\} \mid x \in \{\bigcup\{\{y.\pi_1\} \mid y \in Y\}\}\}$ to $\bigcup\{\{(\bigcup\{\{y.\pi_1\} \mid y \in Y\}, z)\} \mid z \in Z\}$ by the second rule in Figure 3, changes the time complexity from $O(|Y| + |Z|)$ to $O(|Z| \cdot |Y|)$.

Fortunately, the second rule in Figure 3 is the only rule that misbehaves this way. For convenience of reference, the system of rewrite rules in Figure 3 is called the unrestricted system. And when the second rule is excluded, it is called the restricted system.

▶ **Proposition 6** (Normal form). *Let $e(\vec{X}) : s$ be an expression in $\mathcal{NRC}(\leq)$, and $\vec{C}$ be objects having the same types as $\vec{X}$.*

1. *$e[\vec{C}/\vec{X}] == e'[\vec{C}/\vec{X}]$ if $e \mapsto e'$.*
2. *$step(e[\vec{C}/\vec{X}] \Downarrow) \geq step(e'[\vec{C}/\vec{X}] \Downarrow)$ if $e \mapsto e'$.*
3. *$time(e[\vec{C}/\vec{X}] \Downarrow) \geq time(e'[\vec{C}/\vec{X}] \Downarrow)$ if $e \mapsto e'$ under the restricted system of rewrite rules.*
4. *The (un)restricted system of rewrite rules is strongly normalizing.*
5. *The unrestricted system of rewrite rules induces a normal form, wherein every subexpression of the form $\bigcup\{e_1(y, \vec{x}, \vec{X}) \mid y \in e_2(\vec{x}, \vec{X})\}$, $e_2(\vec{x}, \vec{X})$ must be one of the variables in $\vec{X}$.*
6. *The restricted system of rewrite rules induces a normal form, wherein every subexpression of the form $\bigcup\{e_1(y, \vec{x}, \vec{X}) \mid y \in e_2(\vec{x}, \vec{X})\}$, $e_2(\vec{x}, \vec{X})$ must be one of the variables in $\vec{X}$ or $e_2(\vec{x}, \vec{X})$ has the form $\{e_3(\vec{x}, \vec{X})\}$.*

3 A limited-mixing lemma

An analysis of the normal form induced by the restricted system of rewrite rules yields a useful limited-mixing lemma on $\mathcal{NRC}_1(\leq)$. The lemma is proved below, after some relevant definitions are given.

A level-0 atom of an object C is a constant c which has at least one occurrence in C that is not inside any elist in C. A level-1 atom of an object C is a constant c which has at least one occurrence in C that is inside an elist which is not nested inside another elist in C. All other constants appearing in an object C are higher level atoms. The notations $\mathit{atom}^0(C)$, $\mathit{atom}^1(C)$, and $\mathit{atom}^{\leq 1}(C)$ respectively denote the set of level-0 atoms of C, the set of level-1 atoms of C, and their union. The level-0 molecules of an object C are the elists in C that are not nested inside other elists. The notation $\mathit{molecule}^0(C)$ denotes the set of level-0 molecules of C. E.g., suppose $C = (c_1, c_2, \{(c_3, c_4, \{(c_5, c_6)\})\})$; then $\mathit{atom}^0(C) = \{c_1, c_2\}$, $\mathit{atom}^1(C) = \{c_3, c_4\}$, $\mathit{atom}^{\leq 1}(C) = \{c_1, c_2, c_3, c_4\}$, $\{c_5, c_6\}$ are higher-level atoms, and $\mathit{molecule}^0(C) = \{\{(c_3, c_4, \{(c_5, c_6)\})\}\}$.

The level-0 Gaifman graph of an object C is defined as an undirected graph $\mathit{gaifman}^0(C)$ whose nodes are the level-0 atoms of C, and edges are all the pairs of level-0 atoms of C. The level-1 Gaifman graph of an object C is defined as an undirected graph $\mathit{gaifman}^1(C)$ whose nodes are the level-1 atoms of C, and the edges are defined as follow: If $C == \{C_1$, ..., $C_n\}$, the edges are pairs (x, y) such that x and y are in the same $\mathit{atom}^0(C_i)$ for some $1 \leq i \leq n$; if $C == (C_1$, ..., $C_n)$, the edges are pairs $(x, y) \in \mathit{gaifman}^1(C_i)$ for some $1 \leq i \leq n$; and there are no other edges. The Gaifman graph [8] of an object C is defined as $\mathit{gaifman}(C) = \mathit{gaifman}^0(C) \cup\ \mathit{gaifman}^1(C)$.

It is shown below, by induction on the structure of $\mathcal{NRC}_1(\leq)$ expressions, that they manipulate their inputs in highly restricted local manners. In particular, expressions which have contant time complexity are unable to mix level-0 and level-1 atoms. And expressions which have linear time complexity are able to mix level-0 atoms with level-0 and level-1 atoms, but are unable to mix level-1 atoms with themselves or with higher-level atoms.

▶ **Lemma 7** (Limited mixing). *Let $e(\vec{X}) : s$ be an expression in $\mathcal{NRC}_1(\leq)$. Suppose objects $\vec{C}$ have the same types as $\vec{X}$, and $e[\vec{C}/\vec{X}] \Downarrow C'$.*

1. *If $e(\vec{X})$ has constant time complexity, then*
   (i) $\mathit{atom}^0(C') \subseteq \mathit{atom}^0(\vec{C})$,
   (ii) $\mathit{atom}^1(C') \subseteq \mathit{atom}^{\leq 1}(\vec{C})$,
   (iii) $\mathit{gaifman}(C') \subseteq \mathit{gaifman}(\vec{C})$,
   (iv) for each $U \in \mathit{molecule}^0(C')$, there are V_0, V_1, ..., V_m such that $\mathit{atom}^1(V_0) \subseteq \mathit{atom}^0(\vec{C})$, $V_j \in \mathit{molecule}^0(\vec{C})$ for each $1 \leq j \leq m$, and $U = V_0 \cup V_1 \cup \cdots \cup V_m$.
2. *If $e(\vec{X})$ has linear time complexity, then*
   (i) $\mathit{atom}^0(C') \subseteq \mathit{atom}^0(\vec{C})$,
   (ii) $\mathit{atom}^1(C') \subseteq \mathit{atom}^{\leq 1}(\vec{C})$, and
   (iii) for each $(u, v) \in \mathit{gaifman}(C')$, either $(u, v) \in \mathit{gaifman}(\vec{C})$, or $u \in \mathit{atom}^0(\vec{C})$ and $v \in \mathit{atom}^1(\vec{C})$, or $u \in \mathit{atom}^1(\vec{C})$ and $v \in \mathit{atom}^0(\vec{C})$.

Proof. The proof proceeds by structural induction on $e(\vec{X})$. For Part 1, the only interesting case is when $e(\vec{X})$ has constant time complexity and has the form $\bigcup\{e_1(x, \vec{X}) \mid x \in e_2(\vec{X})\}$. This implies both $e_1(x, \vec{X})$ and $e_2(\vec{X})$ have constant time complexity. Let $\vec{C}$ have the types of $\vec{X}$, $e[\vec{C}/\vec{X}] \Downarrow C'$, and $e_2[\vec{C}/\vec{X}] \Downarrow C''$. Then by induction hypothesis on $e_2(\vec{X})$, $\mathit{atom}^0(C'') \subseteq \mathit{atom}^0(\vec{C})$, $\mathit{atom}^1(C'') \subseteq \mathit{atom}^{\leq 1}(\vec{C})$, and $\mathit{gaifman}(C'') \subseteq \mathit{gaifman}(\vec{C})$. Note that $\mathit{molecule}^0(C'') = \{C''\}$. The induction hypothesis implies $C'' = V_0\ \cup\ V_1\ \cup\ \cdots\ \cup\ V_m$ where $\mathit{atom}^1(V_0) \subseteq \mathit{atom}^0(\vec{C})$ and $V_j \in \mathit{molecule}^0(\vec{C})$ for each $j > 0$. This means each V_j, $j > 0$, is one of the input relations in $\vec{C}$. However, this leads to a contradiction because $\bigcup\{e_1(x, \vec{X}) \mid x \in e_2(\vec{X})\}$ would then have at least linear time complexity. So, there can be no V_j, $j > 0$. Hence, $C'' = V_0$ and $\mathit{atom}^1(C'') = \mathit{atom}^1(V_0) \subseteq \mathit{atom}^0(\vec{C})$. Let $C'' = \{C_1, \ldots, C_n\}$. Then, $\mathit{atom}^0(C_i) \subseteq \mathit{atom}^0(\vec{C})$. Let $e_1[C_i/x, \vec{C}/\vec{X}] \Downarrow C'_i$. Then, by induction hypothesis on $e_1(x, \vec{X})$, $\mathit{atom}^0(C'_i) = \{\} \subseteq \mathit{atom}^0(C_i, \vec{C}) = \mathit{atom}^0(\vec{C})$, $\mathit{atom}^1(C'_i) \subseteq \mathit{atom}^{\leq 1}(\vec{C})$, $\mathit{gaifman}(C'_i) \subseteq \mathit{gaifman}(C_i, \vec{C}) = \mathit{gaifman}(\vec{C})$. The induction hypothesis also implies $C'_i = V_{i,0} \cup V_{i,1} \cup \cdots \cup V_{i,m}$ for some m where $\mathit{atom}^0(V_{i,0}) \subseteq \mathit{atom}^0(C_i, \vec{C}) = \mathit{atom}^0(\vec{C})$, and $V_{i,j} = \mathit{molecule}^0(C_i, \vec{C}) = \mathit{molecule}^0(\vec{C})$ for $j > 0$. Thus, each C'_i satisfies Part 1(i) to Part 1(iv). Consequently, $C' = C'_1 \cup \cdots \cup C'_n$ satisfies Part 1(i) to Part 1(iv). The other cases for Part 1 are straightforward, and are thus omitted.

For Part 2, by Proposition 6, $e(\vec{X})$ is assumed to be in the normal form induced by the restricted system of rewrite rules. This first interesting case is when $e(\vec{X})$ has the form $\bigcup\{e_1(x, \vec{X}) \mid x \in X_0\}$, where X_0 is one of the free variables in $\vec{X}$, and has linear time complexity. Then $e_1(x, \vec{X})$ must have constant time complexity; otherwise, the whole expression has quadratic time complexity. Let $\vec{C}$ have the types of $\vec{X}$ and let C_0 in $\vec{C}$ correspond to X_0. Suppose $C_x \in C_0$ and $e_1[C_x/x, \vec{C}/\vec{X}] \Downarrow C'_x$. As C'_x has set type, $\mathit{atom}^0(C'_x) = \{\} \subseteq \mathit{atom}^0(\vec{C})$. This proves Part 2(i). Since $C_x \in C_0$ and C_0 is in $\vec{C}$, $\mathit{atom}^0(C_x) \in \mathit{atom}^1(\vec{C})$. Also, as this lemma concerns $\mathcal{NRC}_1$, C_x must have type $b \times \cdots \times b$; thus, $\mathit{atom}^1(C_x) = \{\}$. By the induction hypothesis on $e_1(x, \vec{X})$, $\mathit{atom}^1(C'_x) \subseteq \mathit{atom}^{\leq 1}(C_x, \vec{C}) = \mathit{atom}^{\leq 1}(\vec{C})$. This proves Part 2(ii). As $e_1(x, \vec{X})$ has constant time complexity, and C_x has type $b \times \cdots \times b$, the induction hypothesis also implies $\mathit{gaifman}(C'_x) \subseteq \mathit{gaifman}(C_x, \vec{C}) = \mathit{gaifman}^0(C_x, \vec{C}) \cup \mathit{gaifman}^1(\vec{C})$. Suppose $(u, v) \in \mathit{gaifman}(C'_x)$. If $u \in \mathit{atom}^0(C_x)$ and $v \in \mathit{atom}^0(C_x)$, then $(u, v) \in \mathit{gaifman}^1(\vec{C}) \subseteq \mathit{gaifman}(\vec{C})$. If $u \in \mathit{atom}^0(C_x)$ and $v \notin \mathit{atom}^0(C_x)$, then $u \in \mathit{atom}^1(C_0) \subseteq \mathit{atom}^1(\vec{C})$ and $v \in \mathit{atom}^0(\vec{C})$. If $u \notin \mathit{atom}^0(C_x)$ and $v \in \mathit{atom}^0(C_x)$, then $v \in \mathit{atom}^1(C_0) \subseteq \mathit{atom}^1(\vec{C})$ and $u \in \mathit{atom}^0(\vec{C})$. If $u \notin \mathit{atom}^0(C_x)$ and $v \notin \mathit{atom}^0(C_x)$, then both u and v are in $\mathit{atom}^0(\vec{C})$, and thus $(u, v) \in \mathit{gaifman}^0(\vec{C}) \subseteq \mathit{gaifman}(\vec{C})$. This proves Part 2(iii). This finishes the case when $e(\vec{X})$ has the form $\bigcup\{e_1(x, \vec{X}) \mid x \in X_0\}$,

The second interesting case is when $e(\vec{X})$has linear time complexity and has the form $\bigcup\{e_1(x, \vec{X}) \mid x \in \{e_2(\vec{X})\}\}$. Let $\vec{C}$ have the types of $\vec{X}$ and let $e_2[\vec{C}/\vec{X}] \Downarrow C''$. By the induction hypothesis of either Part 1 or 2 (it does not matter which), we get $\mathit{atom}^0(C'') \subseteq \mathit{atom}^0(\vec{C})$; thus, $\mathit{atom}^0(C'', \vec{C}) = \mathit{atom}^0(\vec{C})$. Since $\{e_2(\vec{X})\}$ has flat relation type, $e_2(\vec{X})$ must have a type of the form $b \times \cdots \times b$. This means $\mathit{atom}^1(C'') = \{\} \subseteq \mathit{atom}^1(\vec{C})$; thus, $\mathit{atom}^1(C'', \vec{C}) = \mathit{atom}^1(\vec{C})$. Crucially, $\mathit{atom}^0(C'') \subseteq \mathit{atom}^0(\vec{C})$ and $\mathit{atom}^1(C'') = \{\}$ implies $\mathit{gaifman}(C'', \vec{C}) = \mathit{gaifman}(\vec{C})$. As C'' has no elist, $\mathit{molecule}^0(C'', \vec{C}) = \mathit{molecule}^0(\vec{C})$. Then both Part 1 and 2 of the lemma follows immediately for this case.

The other cases are straightfoward and are omitted. ◀

4 Intensional expressiveness gap

As mentioned earlier, an intensional expressiveness gap of comprehension syntax relative to relational database systems appears to manifest in joins of low selectivity. And judging by Example 1, it also potentially manifests in relational intersection and relational difference, as these two operations have $O(n \log n)$ time complexity in a relational database system whereas their comprehension-syntax equivalent in Example 1 is quadratic. The other relational query operations (project, select, and union), as well as joins of high selectivity are succinctly expressible in $\mathcal{NRC}_1(\leq)$ with comparable time complexity when there are no indices available on the input relations; cf. Example 1. The relational division is ignored here because it is not directly supported by typical relational database systems; i.e., when it is needed in a relational database system, it is expressed using the other operators, usually at quadratic space and time complexity [11].

This intensional expressiveness gap is illustrated and confirmed here using two example queries on objects in canonical form. The first query, $\mathit{head}(x, X)$, produces the first element in an input canonical elist X, assuming this first element has the form (x, x') and x does not appear in subsequent elements of X. The second query, $\mathit{zip}(X, Y)$, produces an elist that pairs the ith elements in two input canonical elists X and Y of equal length, assuming the

ith element of X has the form (o_i, x'_i) and that in Y has the form (o_i, y'_i) and that each o_i occurs only once in X and once in Y. These two queries are chosen because *head* can be straightforwardly implemented in constant time in any programming language, while *zip* is a very low-selectivity join which can be answered efficiently – i.e. with linear or near-linear time complexity – in relational database systems.

The expression $\mathit{head}'(x, X) =_{df} \{(y, y') \mid (y, y') \in X, y = x\}$ in $\mathcal{NRC}_1(\leq)$ defines the same function as *head* on any input (x, X) meeting the requirement of *head*. However, $\mathit{head}'(x, X)$ has time complexity $\Theta(|X|)$; i.e., it has linear time complexity.

The expression $\mathit{zip}'(X, Y) =_{df} \{(x, y) \mid (u, x) \in X, (v, y) \in Y, u = v\}$ in $\mathcal{NRC}_1(\leq)$ defines the same function as *zip* on any input (X, Y) meeting the requirement of *zip*. However, $\mathit{zip}'(X, Y)$ has time complexity $\Theta(|X| \cdot |Y|)$; i.e., it has quadratic time complexity.

In fact, as shown below, every expression in $\mathcal{NRC}_1(\leq)$ that implements *head* has at least linear time complexity; and every expression in $\mathcal{NRC}_1(\leq)$ that implements *zip* has at least quadratic time complexity. In other words, the intensional expressiveness gap of $\mathcal{NRC}_1(\leq)$, and thus of comprehension syntax, is real.

▶ **Proposition 8.** *Let* $\mathit{head}(x, X) : \{b_1 \times b_2\}$ *be an expression in* $\mathcal{NRC}_1(\leq)$*. Suppose for every object* c *of type* b_1 *and non-empty canonical object* C *of type* $\{b_1 \times b_2\}$ *whose first element is* (c, c_0)*, and* c *does not appear in subsequent elements of* C*,* $\mathit{head}[c/x, C/X] \Downarrow \{(c, c_0)\}$*. Then* $\mathit{time}(\mathit{head}[c/x, C/X] \Downarrow)$ *is at least* $|C|$*. That is, the time complexity of* $\mathit{head}(x, X)$ *is* $\Omega(|X|)$*.*

Proof. For a contradiction, suppose $\mathit{head}(x, X)$ has sublinear time complexity. Then Proposition 5 implies $\mathit{head}(x, X)$ has constant time complexity. Let $\mathit{head}[c/x, C/X] \Downarrow C'$ where $C' = \{(c, c_0)\}$. As C' has type $\{b_1 \times b_2\}$, $\mathit{molecule}^0(C') = \{C'\}$. Similarly, $\mathit{molecule}^0(c, C) = \{C\}$. By Part 1(iv) of Lemma 7, either $C \subseteq C'$ or $\mathit{atom}^1(C') \subseteq \mathit{atom}^0(c, C)$. However, $C \not\subseteq C' = \{(c, c_0)\}$ in general and $\mathit{atom}^1(C') = \{c, c_0\} \not\subseteq \mathit{atom}^0(c, C) = \{c\}$. This contradiction implies that $\mathit{head}(x, X)$ has at least linear time complexity. ◀

▶ **Proposition 9.** *Let* $\mathit{zip}(X, Y) : \{b_1 \times b_2\}$ *be an expression in* $\mathcal{NRC}_1(\leq)$ *where* X *is a variable of type* $\{b_3 \times b_1\}$*,* Y *is a variable of type* $\{b_3 \times b_2\}$*, and* b_1*,* b_2*, and* b_3 *are distinct base types. Suppose for every canonical objects* $U == \{(o_1, u_1), ..., (o_n, u_n)\}$ *of type* $\{b_3 \times b_1\}$ *and* $V == \{(o_1, v_1), ..., (o_n, v_n)\}$ *of type* $\{b_3 \times b_2\}$*,* $\mathit{zip}[U/X, V/Y] \Downarrow C'$ *where* $C' == \{(u_1, v_1), ..., (u_n, v_n)\}$*. Then* $\mathit{time}(\mathit{zip}[U/X, V/Y] \Downarrow)$ *is at least* $|U| \cdot |V|$*. Thus, the time complexity of* $\mathit{zip}(X, Y)$ *is* $\Omega(|U| * |Y|)$*.*

Proof. Suppose for a contradiction that $\mathit{zip}(X, Y)$ has subquadratic time complexity. Then Proposition 5 implies $\mathit{zip}(X, Y)$ has either constant or linear time complexity.

Assume $\mathit{zip}(X, Y)$ has constant time complexity and $\mathit{zip}[U/X, V/Y] \Downarrow C'$ where $C' == \{(u_1, v_1), ..., (u_n, v_n)\}$. Clearly, $\mathit{molecule}^0(C') = \{C'\}$ and $\mathit{molecule}^0(U, V) = \{U, V\}$. Then, by Part 1(iv) of Lemma 7, either $U \subseteq C'$, $V \subseteq C'$, or $\mathit{atom}^1(C') \subseteq \mathit{atom}^0(U, V) = \{\}$. Clearly, all three options are impossible. Thus $\mathit{zip}(X, Y)$ cannot have constant time complexity.

Suppose instead $\mathit{zip}(X, Y)$ has linear time complexity. Then $\mathit{gaifman}(C') = C' = \{(u_1, v_1), ..., (u_n, v_n)\}$. However, for $1 \leq i \leq n$, $(u_i, v_i) \in \mathit{gaifman}(C') \notin \mathit{gaifman}(U, V) = U \cup V$. Then, by Part 2(iii) of Lemma 7, either $u_i \in \mathit{atom}^0(U, V)$ or $v_i \in \mathit{atom}^0(U, V)$. However, as U and V are both elists, $\mathit{atom}^0(U, V) = \{\}$ and thus contains neither u_i nor v_i. So, $\mathit{zip}(X, Y)$ cannot have linear time complexity. Therefore, it has at least quadratic time complexity. ◀

Incidentally, it was Peter Buneman who first conjectured that *zip*, characterized by its low-selectivity nature, could only be defined using comprehension syntax with quadratic time complexity. This insight, shared with me by Stijn Vansummeren over three decades ago, has apparently remained open until being resolved here in Proposition 9.

5 Closing remarks

The impedance mismatch problem between databases and programming languages has been highlighted three decades ago [7]. It refers to the difficulties of integrating database query-like feature and capability into a programming language. Some has regarded the use of comprehension syntax [3] as a breakthrough for this problem [6]. Indeed, comprehension syntax provides an iteration construct that is simple enough for programming with collection data types that data objects of a database have been mapped to, and explicit enough to admit a direct translation to the query language of the database, thereby permitting queries to the database to be embedded simply and naturally into a programming language.

However, comprehension syntax is also widely adopted in modern programming languages – e.g., Python [9] and Scala [16] – as an easy-to-use means for manipulating collection types in general. For this purpose, the collection objects are created within a program or do not come from a database system, and queries written in comprehension syntax for manipulating these objects are not translated to the query language of an underlying database system for execution. In such a setting, programs written in comprehension syntax typically correspond to nested loops.

This gives rise to an intriguing disparity. Many queries when translated to their database equivalent can be executed by the underlying database system very efficiently. Yet when they are executed directly as comprehension syntax, they are not efficient at all. Consider this query as an example, $\{(x.dept, x.stf) \mid x \in \mathit{DeptStaff}, \, y \in \mathit{Staff}, \, x.stf = y.stf, \, y.age > 65\}$ which retrieves departments and their staff who are above 65 years old. Suppose a staff typically belongs to only one department. This query would then be a low-selectivity join. It typically would be executed by a database system, via e.g. a merge join [2], with time complexity $\Theta(n+m)$ assuming the inputs *DeptStaff* and *Staff* have size n and m and are both sorted by their *stf* field; or with time complexity $\Theta(n \log(n) + m \log(m))$ if sorting is required. In contrast, the same query would typically has time complexity $\Theta(nm)$ natively in the programming language. Even if a filter promotion is applied (and ignoring the change in the appearance of the output) to optimize the query to $\{(x.dept, x.stf) \mid y \in \mathit{Staff}, \, y.age > 65, \, x \in \mathit{DeptStaff}, \, x.stf = y.stf\}$, this optimized query still has quadratic time complexity $\Theta(gnm)$, for some $0 \leq g \leq 1$, natively in the programming language.

This linear-vs-quadratic time complexity difference of low-selectivity joins can be called an intensional expressiveness gap between comprehension syntax and database systems. That is, it is a gap between the algorithms that can be expressed using comprehension syntax and database systems. As far as relational database system is concerned, the low-selectivity join, the relational intersection, and the relational difference appear to be the only intensional expressiveness gap as all other relational query operators, as well as high-selectivity joins, in the absence of database indices on the input relations, have similar time complexity whether executed by a relational database system or in the programming language directly as queries in comprehension syntax.

It has been open whether this intensional expressiveness gap is a real gap; i.e., there might exist some clever way to implement low-selectivity joins efficiently using comprehension syntax. As the main result of this paper, this intensional expressiveness gap is proved by showing that all subquadratic algorithms expressible using pure comprehension syntax cannot compute low-selectivity joins. In fact, I have claimed elsewhere [17] that even allowing some functions – viz. *takewhile* and *dropwhile*, *fold*, or *zip* – commonly available in the collection-type function libraries of programming languages, to be used with comprehension syntax, all expressible subquadratic algorithms still cannot compute low-selectivity joins in general.

It is a natural follow-up question on what exactly is missing from comprehension syntax that prevents efficient algorithms for low-selectivity joins to be expressed. This intensional expressiveness gap can be charaterized in a precise way by identifying a new programming construct that enables more algorithms to be expressed but doing so without enabling more functions to be expressed. This is the Synchrony iterator construct, which I have proposed and investigated with Val Tannen and Stefano Perna [17], for expressing synchronized iterations on multiple collection objects. A construct for generalized iteration on multiple collection objects in synchrony appears to be a conceptually novel choice, because practically all functions commonly provided in the function libraries of programming languages involve iteration on a single collection object. Adding this construct does not change the functions that are expressible using pure comprehension syntax, and yet enables the realization of efficient low-selectivity joins, including non-equijoins. Moreover, the Synchrony iterator construct dovetails rather appealingly with comprehension syntax, so that efficient queries written with the help of Synchrony iterators often do not look too different from their inefficient pure comprehension-syntax equivalents. See [17] for more information.

The proof of the intensional expressiveness gap uses a novel limited-mixing lemma. The lemma shows that all subquadratic-time queries in comprehension syntax are only able to mix atomic objects in their input in very limited ways. This limited-mixing lemma is of independent interest. Many past works on intensional expressive power are query specific. Just to cite a couple of examples, Abiteboul and Vianu [1] showed that there is no "generic machine" for computing the parity query in PTIME; and Suciu and Paredaens [18] showed that the transitive closure of a long chain can only be computed in the complex object algebra of Abiteboul and Beeri using exponential space. A notable non-query-specific intensional expressiveness result is that of Wong [24], who showed that all queries on a general class of structures, which includes deep trees and long chains, in a nested relational calculus augmented with a powerset operator are either already expressible in the calculus without using the powerset operator, or must use an exponential amount of space. Furthermore, most previous results on intensional expressive power, such as those mentioned above, are for query languages without ordered data types. The limited-mixing lemma in this paper stands out in comparison to these results in two aspects. Firstly, the limited-mixing lemma is non-query specific; it applies to all queries of subquaratic time complexity in the respective query languages. Secondly, the limited-mixing lemma is valid in the presence of ordered data types. The limited-mixing lemma thus enriches the repertoire of techniques for studying intensional expressive power. The limited-mixing lemma is also useful intensional counterpart to Gaifman's locality property [8]. Gaifman's locality property is useful for analyzing the extensional and intensional expressive power [10, 12, 24] of query languages on unordered data types. However, it is effectively useless on ordered data types and on query languages with a *fold*-like function. Limited-mixing lemmas do not have these limitations.

Lastly, here is a small advertisement: Synchrony iterator has been implemented in Python and Scala. These implementations are available at `https://www.comp.nus.edu.sg/~wongls/projects/synchrony`.

References

1 Serge Abiteboul and Victor Vianu. Generic computation and its complexity. In *Proceedings of 23rd ACM Symposium on the Theory of Computing*, pages 209–219, 1991.

2 Michael W. Blasgen and Kapali Eswaran. Storage and access in relational databases. *IBM Systems Journal*, 16(4):363–377, 1977.

3 Peter Buneman, Leonid Libkin, Dan Suciu, Val Tannen, and Limsoon Wong. Comprehension syntax. *SIGMOD Record*, 23(1):87–96, March 1994.

4 Peter Buneman, Shamim Naqvi, Val Tannen, and Limsoon Wong. Principles of programming with complex objects and collection types. *Theoretical Computer Science*, 149(1):3–48, September 1995.

5 Edgar F. Codd. Relational completeness of data base sublanguages. In R. Rustin, editor, *Data Base Systems*, pages 65–98. Prentice-Hall, 1972.

6 Ezra Cooper. The script-writer dream: How to write great SQL in your own language, and be sure it will succeed. In *Proceedings of 12th International Symposium on Database Query Languages*, pages 36–51, Lyon, France, August 2009.

7 George Copeland and David Maier. Making Smalltalk a database system. In *Proceedings of ACM-SIGMOD 84*, pages 316–325, Boston, MA, June 1984.

8 Haim Gaifman. On local and non-local properties. In *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pages 105–135. North Holland, 1982.

9 John V. Guttag. *Introduction to Computation and Programming Using Python: With Application to Understanding Data.* MIT Press, 2016.

10 Lauri Hella, Leonid Libkin, and Juha Nurmonen. Notions of locality and their logical characterizations over finite models. *Journal of Symbolic Logic*, 64(4):1751–1773, 1999.

11 Dirk Leinders and Jan Van den Bussche. On the complexity of division and set joins in the relational algebra. *Journal of Computer and System Sciences*, 73(4):538–549, June 2007.

12 Leonid Libkin. On the forms of locality over finite models. In *Proceedings of 12th IEEE Symposium on Logic in Computer Science*, pages 204–215, 1997.

13 Leonid Libkin and Limsoon Wong. Aggregate functions, conservative extension, and linear orders. In Catriel Beeri, Atsushi Ohori, and Dennis E. Shasha, editors, *Proceedings of 4th International Workshop on Database Programming Languages, New York, August 1993*, pages 282–294. Springer-Verlag, January 1994.

14 Leonid Libkin and Limsoon Wong. Conservativity of nested relational calculi with internal generic functions. *Information Processing Letters*, 49(6):273–280, March 1994.

15 Leonid Libkin and Limsoon Wong. Query languages for bags and aggregate functions. *Journal of Computer and System Sciences*, 55(2):241–272, October 1997.

16 Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala: A Comprehensive Step-by-Step Guide.* Artima Inc., December 2019.

17 Stefano Perna, Val Tannen, and Limsoon Wong. Iterating on multiple collections in synchrony. *Journal of Functional Programming*, 32:e9, July 2022.

18 Dan Suciu and Jan Paredaens. The complexity of the evaluation of complex algebra expressions. *Journal of Computer and Systems Sciences*, 55(2):322–343, October 1997.

19 Dan Suciu and Limsoon Wong. On two forms of structural recursion. In *LNCS 893: Proceedings of 5th International Conference on Database Theory*, pages 111–124, Prague, January 1995. Springer-Verlag.

20 Stan J. Thomas and Patrick C. Fischer. *Nested Relational Structures*, pages 269–307. JAI Press, London, England, 1986.

21 Wolfgang Wechler. *Universal Algebra for Computer Scientists*, volume 25 of *EATCS Monograph on Theoretical Computer Science.* Springer-Verlag, Berlin, 1992.

22 Limsoon Wong. Normal forms and conservative extension properties for query languages over collection types. *Journal of Computer and System Sciences*, 52(3):495–505, June 1996.

23 Limsoon Wong. Kleisli, a functional query system. *Journal of Functional Programming*, 10(1):19–56, 2000.

24 Limsoon Wong. A dichotomy in the intensional expressive power of nested relational calculi augmented with aggregate functions and a powerset operator. In *Proceedings of 32nd ACM Symposium on Principles of Database Systems*, pages 285–295, New York, June 2013.