String Diagrams for Graded Monoidal Theories, with an Application to Imprecise Probability

Ralph Sarkis
Department of Computer Science, University College London, UK

Fabio Zanasi
Department of Computer Science, University College London, UK

Abstract

We introduce string diagrams for graded symmetric monoidal categories. Our approach includes a definition of graded monoidal theory and the corresponding freely generated syntactic category. Also, we show how an axiomatic presentation for the graded theory may be modularly obtained from one for the grading theory and one for the base category. The **Para** construction on monoidal actegories is a motivating example for our framework. As a case study, we show how to axiomatise a variant of the graded category **ImP**, recently introduced by Liell-Cock and Staton to model imprecise probability [34]. This culminates in a representation, as string diagrams with grading wires, of programs with primitives for nondeterministic and probabilistic choices and conditioning.

2012 ACM Subject Classification Theory of computation → Equational logic and rewriting; Mathematics of computing → Probabilistic representations

Keywords and phrases string diagrams, graded categories, probability, nondeterminism

Digital Object Identifier 10.4230/LIPIcs.CALCO.2025.5

Related Version *Full Version*: `https://arxiv.org/abs/2501.18404`

Funding This research was supported by the ARIA Safeguarded AI programme.

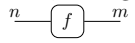
1 Introduction

Graded categorical structures have been successfully applied in the study of effect systems [38, 30, 24, 40, 47], coalgebraic semantics [39, 18, 20, 19, 21], program logics [25], cellular automata [12], and most recently to model imprecise probabilistic reasoning [34].

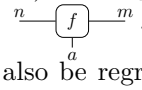
In a graded category $\mathbf{C}$, every morphism f has a grade γ that is an object in a monoidal category $\mathbb{G}$. This grade can be understood as a parameter that f takes into account separately from its inputs and outputs. Grades have been used to represent side-effects of a program [30, 25], a bound on the length of relevant traces [39], the neihbourhood of a cell in a cellular automaton [12], or the number of nondeterministic choices consumed by a process [34]. When composing graded morphisms, in sequence or in parallel, the grades combine according to the monoidal product in $\mathbb{G}$. The grade of a morphism can also be changed by regrading with a morphism in $\mathbb{G}$, which informally consists in acting on parameters before the graded morphism has access to them.

Somewhat independently from these developments, there is an increasingly popular research area using monoidal categories to model computational processes that are attentive to resources, such as those found in probabilistic programming, quantum theory, concurrency, and machine learning. This family of approaches adopts *string diagrams*, the graphical language of monoidal categories [48, 44], as a visual and yet rigorous syntax for processes, providing an intuitive understanding of how information flows and is exchanged between components within a system. Particular emphasis is put on finding an *axiomatic presentation* of monoidal categories of processes, in the form of a diagrammatic calculus whose equational theory is complete with respect to the semantic equivalence of interest. This perspective

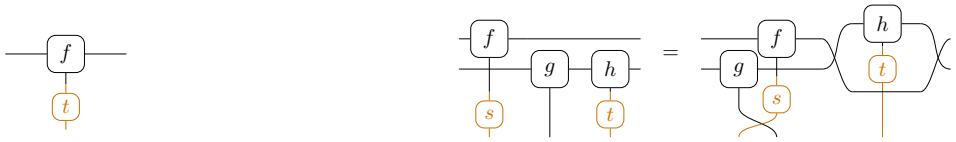
has been fruitful for instance in quantum theory [27, 28, 29, 2, 46], concurrency [8], linear algebra [53, 3, 4], automata theory [43, 1], and probabilistic programming [42, 49]. It has determined the development of *monoidal algebra* [10], as the counterpart of traditional categorical algebra [33] adapted to a monoidal rather than cartesian setting.

Given how much graded category theory and monoidal algebra overlap in terms of applications, it is natural to ask how they can be combined in a uniform approach. Even though many extensions of monoidal algebra have been considered [9, 17, 6, 7, 5, 50, 35], none of these approaches readily incorporates a notion of grade. In essence, string diagrams always represent a morphism as a box with input and output wires 

, whereas morphisms in a graded category also interact through a grade, which cannot be conflated with the domain or codomain. The syntax for bicategories [45] and double categories [41] *does* allow for another axis of sequential composition orthogonal to the domain/codomain one. However, this makes it difficult to represent parallel composition, and the link between graded categories and 2-dimensional categories is yet to be fully understood.

To fill this gap, we propose a string diagrammatic language for graded symmetric monoidal categories (SMC). A morphism is pictured as a box with input wires, output wires, and grading wires: 

. Morphisms can be composed in sequence and in parallel as usual, but they can also be regraded. We represent this last operation by plugging a diagram representing the grading morphism to the grading wire as shown below on the left. Translating the laws of graded SMCs to this graphical syntax (Figure 1) shows that any reasonable deformation of a graded string diagram does not change the represented morphism, e.g. the two diagrams below on the right are equal.



Given a signature where operation symbols have an arity, a coarity, and a grade, we show how to freely generate a graded SMC of string diagrams that obey these visually intuitive equations. We also show how to impose further equations that may be necessary to accurately model some concrete setting. This development closely follows the analogue for (classical) monoidal theories, and in fact we prove (Lemma 24) that diagrams with grade 0 (i.e. no grading wires) can actually be considered as classical string diagrams over the signature restricted to operations with grade 0.

Our syntax draws inspiration from similar pictorial languages for categories obtained with the **Para** construction [14, 16, 26], which however were not formalised. In fact, our framework directly applies to the **Para** construction, and provides the following "modularity" principle (Theorem 27): given axiomatisations of SMCs $\mathbb{G}$ and $\mathbf{C}$, and a suitable action of $\mathbb{G}$ on $\mathbf{C}$, we can always derive an axiomatisation of the graded category $\mathbf{Para}(\mathbb{G}, \mathbf{C})$.

We use this result to axiomatise a graded string diagrammatic language that models imprecise probability following Liell-Cock and Staton [34]. We apply the **Para** construction as they do, but on a subcategory of **FinStoch** that was recently axiomatised in [42], to obtain the graded category **BImP**. Morphisms $n \to m$ of grade a in **BImP** are stochastic matrices divided in 2^a submatrices of dimension $2^m \times 2^n$. Intuitively, such a morphism views each submatrix as a stochastic process and nondeterministically chooses one of them to execute. Regrading in **BImP** amounts to reordering and copying submatrices (not all such operations are possible).

The main result in [42] states that the category of stochastic matrices whose dimensions are powers of 2 is presented by a theory they call CausCirc. Following the modular recipe given in Theorem 27, we add a generator ⌜ of grade 1 to CausCirc that we interpret as outputing a value which nondeterministically evaluates to true or false, and an equation saying that to delete this value is the same thing as not outputing it. We obtain the graded theory $\mathsf{CausCirc}_i$ (imprecise CausCirc), which axiomatises the graded category **BImP**.

We can thus extend the programming language in [42], which is interpreted in CausCirc, with a construct for nondeterministic choice, denoted with `knight`, and interpret it in $\mathsf{CausCirc}_i$. We show (Theorem 33) this language is commutative and affine, meaning that expressions that are indepedent of each other (e.g. they concern different variables) can be rearranged. Furthermore, thanks to our modular approach, we can easily add the conditioning generator (and its corresponding programming construct) studied in [42]. We illustrate the features of this graphical programming language with two probabilistic riddles that are represented as string diagrams (Example 35).

Synopsis. In Section 2, we recall the necessary background on monoidal theories and actegories. Section 3 contains the main definitions of this work. We introduce here graded SMCs, graded monoidal theories, and their string diagrammatic syntax. We also describe the construction of a syntactic graded prop generated by a graded monoidal theory, thereby providing a sound and complete list of equational laws to reason with terms of such theories. In Section 4, we show how to combine two monoidal theories into a graded monoidal theories through the graded Para construction. In Section 5, we define the graded category **BImP** and we apply the result of Section 4 to obtain a presentation. Finally, in Section 6, we obtain a graphical interpretation of a programming language with nondeterministic and probabilistic constructs, illustrating our approach with two concrete examples. The appendix contains complete proofs of our results.

2 Background

We recall background on string diagrams and monoidal theories, following the terminology of [10]. First, a **prop** is a symmetric strict monoidal category (SMC) whose objects are the natural numbers and whose monoidal product on objects is addition. Props are the basic framework to study algebraic objects with a monoidal structure. They may be freely obtained from generators and equations. This requires introducing a notion of symmetric monoidal signature and symmetric monoidal theory. We will omit "symmetric" for brevity.

▶ **Definition 1** (Monoidal term). *A* ***monoidal signature*** Σ *is a set of* ***generators****, each with an* ***arity*** n *and* ***coarity*** m *in* $\mathbb{N}$*. We use the notation* $g : n \to m \in \Sigma$ *to compactly represent the data of a generator in the signature. We inductively define the set* $\mathcal{M}_\Sigma$ *of* ***monoidal*** Σ***-terms*** *(and their (co)arities) built with the generators in* Σ *via the rules* (1)–(4)*.*

$$\frac{g : n \to m \in \Sigma}{g : n \to m \in \mathcal{M}_\Sigma} \quad (1) \qquad \frac{}{\mathrm{id}_0 : 0 \to 0 \in \mathcal{M}_\Sigma} \quad \frac{}{\mathrm{id}_1 : 1 \to 1 \in \mathcal{M}_\Sigma} \quad \frac{}{\sigma_{1,1} : 2 \to 2 \in \mathcal{M}_\Sigma} \quad (2)$$

$$\frac{f : n \to m \in \mathcal{M}_\Sigma \qquad g : m \to \ell \in \mathcal{M}_\Sigma}{f ; g : n \to \ell \in \mathcal{M}_\Sigma} \quad (3) \qquad \frac{f : n \to m \in \mathcal{M}_\Sigma \qquad f' : n' \to m' \in \mathcal{M}_\Sigma}{f \otimes f' : n + n' \to m + m' \in \mathcal{M}_\Sigma} \quad (4)$$

As customary, we use string diagrams *(see e.g. [48, 44]) to visually represent monoidal terms. A generic term* $f : n \to m \in \mathcal{M}_\Sigma$ *is depicted as a box with* n *wires on the left and* m *wires on the right:*  *. We write generator* id_0 *as*  *,* id_1 *as*  *, and* $\sigma_{1,1}$ *as*  *. The two forms of composition are respectively*  ;  := *and*  ⊗  :=  .

▶ **Definition 2** (Theory). *A **monoidal theory** is a tuple $\mathsf{T} = (\Sigma, E)$, where Σ is a monoidal signature, and E is a set of equations, called **axioms**, between monoidal Σ-terms, i.e. pairs $(s,t) \in \mathcal{M}_\Sigma \times \mathcal{M}_\Sigma$ that we denote with $s = t$.*

▶ **Definition 3** (Model). *Let $\mathsf{T} = (\Sigma, E)$ be a monoidal theory, $\mathbf{C}$ an SMC, and $M1$ an object in $\mathbf{C}$. Given an assignment $M : \Sigma \to \mathbf{C}$ sending each generator $g : n \to m$ to a morphism in $\mathbf{C}(M1^{\otimes n}, M1^{\otimes m})$, we can canonically extend M to all terms in $\mathcal{M}_\Sigma$, because each term formation rule (1)–(4) corresponds to an operation in SMCs. A **model** of T valued in $\mathbf{C}$ is such an object $M1$ and assignment M satisfying $M(s) = M(t)$ for all axioms $s = t \in E$.*

Models of monoidal theories are usually viewed through the lens of functorial semantics, *à la* Lawvere [33]. In a nutshell, we can construct a syntactic category $\mathcal{M}_\mathsf{T}$ for a theory T, such that models of T correspond to symmetric strict monoidal functors with domain $\mathcal{M}_\mathsf{T}$. The morphisms of $\mathcal{M}_\mathsf{T}$ are monoidal terms quotiented by some congruence generated by the axioms in T and the necessary equations to satisfy the laws of an SMC.

▶ **Definition 4** (Closure). *Given a theory $\mathsf{T} = (\Sigma, E)$, the **closure** of E, denoted $\overline{E}$, is the smallest congruence on $\mathcal{M}_\Sigma$ containing E and all the laws of SMCs, see Figure A.1.*

▶ **Definition 5** (Syntactic category). *The **syntactic category** of a monoidal theory $\mathsf{T} = (\Sigma, E)$ is a prop $\mathcal{M}_\mathsf{T}$ where the set morphisms $\mathcal{M}_\mathsf{T}(n,m)$ consists of monoidal Σ-terms of arity n and coarity m, quotiented by the congruence $\overline{E}$. Identities, symmetries, sequential composition, and monoidal product are defined via formation rules (1)–(4). The axioms of Figure A.1 ensure that all the laws of SMCs hold.*

*We say that T is a **presentation** of a prop $\mathbf{C}$ (or simply that T presents $\mathbf{C}$) if $\mathcal{M}_\mathsf{T} \cong \mathbf{C}$.*

Functorial semantics is founded upon the fact that models of T valued in $\mathbf{C}$ are in 1-to-1 correspondence with symmetric monoidal functors $\mathcal{M}_\mathsf{T} \to \mathbf{C}$.

The **Para** construction, which plays a role in our developments, is based on *actegories*. Actegories, an abstraction of the concept of monoid/group action, are categories being acted on by monoidal categories. We recall the full definition from [13, 14].

▶ **Definition 6** (Actegory). *Let $\mathbb{G}$ be an SMC and $\mathbf{C}$ be a category. An **action** of $\mathbb{G}$ on $\mathbf{C}$ is a functor $\bullet : \mathbb{G} \times \mathbf{C} \to \mathbf{C}$ equipped with two natural family of isomorphisms, $\eta_X : I \bullet X \cong X$ and $\mu_{A,B,X} : A \bullet (B \bullet X) \cong (A \otimes B) \bullet X$, that satisfy some coherence laws listed in [13, Definition 3.1.1]. We say that $\mathbf{C}$ equipped with this action is a $\mathbb{G}$-**actegory**. We call $\bullet$ a **symmetric monoidal action** and $\mathbf{C}$ a **symmetric monoidal** $\mathbb{G}$-**actegory** if $\mathbf{C}$ is an SMC, $\bullet$ is a symmetric monoidal functor, and there is a natural family of isomorphisms, $\kappa_{A,X,Y} : A \bullet (X \otimes Y) \cong X \otimes (A \bullet Y)$, that satisfy additional coherence laws.*

▶ **Example 7** (Self-action). Any SMC $\mathbf{C}$ acts on itself in a canonical way: $\bullet : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ is defined as the monoidal product $\otimes$ [13, Example 3.2.4]. This is a symmetric monoidal action where the natural isomorphisms η, μ, κ are compositions of coherence isomorphisms. Any subcategory of $\mathbf{C}$ with the inherited SMC structure also acts on $\mathbf{C}$ in the same way.

3 Graded Monoidal Theories and Graded Props

Let $\mathbb{G}$ be a symmetric semicartesian strict monoidal category, i.e. an SMC where the unit is the terminal object. We call the objects of $\mathbb{G}$ **grades** and denote them with γ, ε. We denote composition in $\mathbb{G}$ with ;, monoidal product with juxtaposition, the unit with $\mathbf{1}$, and the symmetries with $\sigma_{\gamma\gamma'} : \gamma\gamma' \to \gamma'\gamma$. Following $\mathbb{G}$-graded categories [52, 36], we introduce the notion of SMC graded over $\mathbb{G}$. In a nutshell, every morphism of a $\mathbb{G}$-graded SMC comes with a grade inside $\mathbb{G}$, and whenever one composes morphisms (in sequence or in parallel), grades combine via the monoidal product in $\mathbb{G}$.

▶ **Definition 8** (Graded SMC). *A $\mathbb{G}$-**graded symmetric strict monoidal category** $\mathbf{C}$ consists of the following data.*

- *A class of **objects** $\mathrm{Ob}(\mathbf{C})$. We write $X \in \mathbf{C}$ instead of $X \in \mathrm{Ob}(\mathbf{C})$.*
- *For all objects $X, Y \in \mathbf{C}$ and grade $\gamma \in \mathbb{G}$, a set $\mathbf{C}_\gamma(X,Y)$ of **morphisms of grade γ**. We often write $f : X \xrightarrow{\gamma} Y$ to mean $f \in \mathbf{C}_\gamma(X,Y)$.*
- *For all objects $X, Y \in \mathbf{C}$ and morphism between grades $t : \gamma \to \varepsilon$, a function $t \star - : \mathbf{C}_\varepsilon(X,Y) \to \mathbf{C}_\gamma(X,Y)$ called **regrading by t**.*
- *For all $X, Y, Z \in \mathbf{C}$ and $\gamma, \varepsilon \in \mathbb{G}$, a **composition** $\fatsemi : \mathbf{C}_\gamma(X,Y) \times \mathbf{C}_\varepsilon(Y,Z) \to \mathbf{C}_{\gamma\varepsilon}(X,Z)$.*
- *A **monoidal product** operation $\otimes : \mathrm{Ob}(\mathbf{C}) \times \mathrm{Ob}(\mathbf{C}) \to \mathrm{Ob}(\mathbf{C})$ and a **unit** $I \in \mathrm{Ob}(\mathbf{C})$.*
- *For all $X, X', Y, Y' \in \mathbf{C}$ and $\gamma, \varepsilon \in \mathbb{G}$, a monoidal product of morphisms defined as a function $\otimes : \mathbf{C}_\gamma(X,Y) \times \mathbf{C}_\varepsilon(X',Y') \to \mathbf{C}_{\gamma\varepsilon}(X \otimes X', Y \otimes Y')$.*
- *For all $X, X' \in \mathbf{C}$, a morphism $\sigma_{X,X'} : X \otimes X' \xrightarrow{1} X' \otimes X$ of grade $\mathbf{1}$ called the **symmetry**.*
- *Composition, monoidal product, and symmetries must satisfy properties akin to those of SMCs, and be compatible with regrading. The full list of conditions is in Section B, but our diagrammatic syntax will help to visualise them (see Figure 1).*

▶ Remark 9. The interchange law cannot hold as usual because the monoidal product in $\mathbb{G}$ is not commutative, and this implies the grades of $(f \otimes f') \fatsemi (g \otimes g')$ and $(f \fatsemi g) \otimes (f' \fatsemi g')$ do not coincide. They differ only by a regrading by a symmetry in $\mathbb{G}$. The law is required to hold up to that regrading. The same reasoning applies to the sliding equation.

▶ Remark 10. The adjective "graded" is overloaded in mathematics, but it usually means that each element of some structure is equipped with an element of another structure (usually a monoid). Some authors prefer to use "locally graded" for the concept we study in this paper. A folklore result (see e.g. [52, 26, 34, 36]) establishes a correspondence between $\mathbb{G}$-graded categories and $[\mathbb{G}^{\mathrm{op}}, \mathbf{Set}]$-enriched categories.

▶ **Definition 11** (Graded functor). *Given two graded SMCs $\mathbf{C}$ and $\mathbf{D}$, a $\mathbb{G}$-**graded symmetric strict monoidal functor** $F : \mathbf{C} \to \mathbf{D}$ is a function $F : \mathrm{Ob}(\mathbf{C}) \to \mathrm{Ob}(\mathbf{D})$ along with a family of functions $F_\gamma : \mathbf{C}_\gamma(X,Y) \to \mathbf{D}_\gamma(FX, FY)$ indexed by $X, Y \in \mathbf{C}$ and $\gamma \in \mathbb{G}$ (although we omit indices in the notation) that preserve the structure of graded SMCs.*

It will be useful to consider the category of $\mathbf{1}$-graded morphisms inside a graded SMC as in e.g. [31, Definition 8.1] and [37, Definition 12].

▶ **Proposition 12.** *Any $\mathbb{G}$-graded SMC $\mathbf{C}$ has an underlying SMC, which we denote with $\mathbf{C_1}$, whose objects are $\mathrm{Ob}(\mathbf{C})$ and morphisms are $\mathbf{1}$-graded morphisms of $\mathbf{C}$.*

▶ **Example 13** (Graded Para). A known method to build a graded category is a variant of the Para construction. We are particularly interested in the Para construction for symmetric monoidal actegories described in [14, Sections 2 and 2.1], but instead of building a bicategory, we build a graded category. Given a semicartesian prop $\mathbb{G}$, an SMC $\mathbf{C}$, and a symmetric monoidal action $\bullet : \mathbb{G} \times \mathbf{C} \to \mathbf{C}$ (Definition 6), there is a $\mathbb{G}$-graded SMC $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ whose underlying SMC is $\mathbf{C}$ and morphisms $X \xrightarrow{a} Y$ are $\mathbf{C}$-morphisms $a \bullet X \to Y$. We defer details of the general construction to Section B as we will mostly be interested in the instantiation $\mathbf{BImP} = \mathbf{Para}(\mathbf{FinInj}^{\mathrm{op}}, \mathbf{BinStoch})$ defined in Section 5.2.

Following the development of monoidal algebra (Section 2), we restrict our focus to graded SMCs whose objects are natural numbers, which are more amenable to axiomatisation.

▶ **Definition 14** (Graded prop). *A $\mathbb{G}$-graded SMC $\mathbf{C}$ is a $\mathbb{G}$-**graded prop** if both $\mathbb{G}$ and $\mathbf{C_1}$ are props. To emphasise that we are working inside a graded prop, we will use the symbols a, b, c for grades, n, m, ℓ for objects, $+$ for monoidal product, and 0 for the terminal object.*

▶ **Definition 15** (Graded term). *A **graded (monoidal) signature** Σ is a set of **generators**, each with an **arity** n, a **coarity** m, and a **grade** a in $\mathbb{N}$. We use the notation $g : n \xrightarrow{a} m \in \Sigma$ to compactly represent the data of a generator in the signature. We inductively define the set $\mathcal{S}_\Sigma$ of **graded (monoidal) Σ-terms** built with the generators in Σ and $\mathbb{G}$-morphisms via the rules (5)–(9). The rules also define the (co)arities and grades of such terms.*

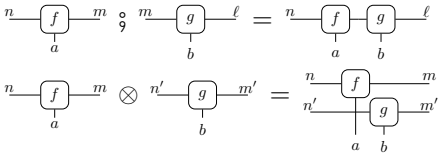
$$\frac{g : n \xrightarrow{a} m \in \Sigma_{\mathsf{h}}}{g : n \xrightarrow{a} m \in \mathcal{S}_\Sigma} \quad (5) \qquad \frac{t : b \to a \in \mathbb{G} \quad f : n \xrightarrow{a} m \in \mathcal{S}_\Sigma}{t \star f : n \xrightarrow{b} m \in \mathcal{S}_\Sigma} \quad (7)$$

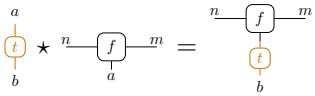
$$\frac{f : n \xrightarrow{a} m \in \mathcal{S}_\Sigma \quad g : m \xrightarrow{b} \ell \in \mathcal{S}_\Sigma}{f \fatsemi g : n \xrightarrow{a+b} \ell \in \mathcal{S}_\Sigma} \quad (6) \qquad \frac{f : n \xrightarrow{a} m \in \mathcal{S}_\Sigma \quad f' : n' \xrightarrow{b} m' \in \mathcal{S}_\Sigma}{f \otimes f' : n + n' \xrightarrow{a+b} m + m' \in \mathcal{S}_\Sigma} \quad (8)$$

$$\frac{}{\mathrm{id}_0 : 0 \xrightarrow{0} 0 \in \mathcal{S}_\Sigma} \qquad \frac{}{\mathrm{id}_1 : 1 \xrightarrow{0} 1 \in \mathcal{S}_\Sigma} \qquad \frac{}{\sigma_{1,1} : 2 \xrightarrow{0} 2 \in \mathcal{S}_\Sigma} \quad (9)$$

These operations on graded terms (simply called terms in the sequel) correspond straightforwardly with the structure of a graded SMC.

We introduce a two-dimensional graphical representation for graded terms as string diagrams. We denote a term $f : n \xrightarrow{a} m$ by a box with n dangling wires on the left (or a single wire labelled with n), m wires on the right, and a wires on the bottom:  . As before, we write generator id_0 as  , id_1 as , and $\sigma_{1,1}$ as . We write sequential composition, monoidal product, and regrading as below. The colouring on $\mathbb{G}$-morphisms is only a reading aid, and it can safely be ignored. Graded terms are read from left to right, then top to bottom, while $\mathbb{G}$-morphisms are read from bottom to top then left to right.





▶ Remark 16. The *grading* wire coming orthogonally to the (co)arity wire is inspired by the string diagrams used for the Para construction in e.g. [14, 16, 26], although our grading wires come from the bottom rather than the top (see Remark 32).

▶ **Definition 17** (Theory). *A **$\mathbb{G}$-graded monoidal theory** is a tuple $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$, where Σ is a graded signature, and E_{h} is a set of equations, called **axioms**, between graded Σ-terms, i.e. pairs $(s, t) \in \mathcal{S}_\Sigma \times \mathcal{S}_\Sigma$ that we denote with $s = t$.*

▶ **Definition 18** (Model). *Let $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$ be a $\mathbb{G}$-graded monoidal theory, $\mathbf{C}$ a $\mathbb{G}$-graded SMC, and $M1$ an object in $\mathbf{C}$. Given an assignment $M : \Sigma \to \mathbf{C}$ sending each generator $g : n \xrightarrow{a} m$ to a morphism in $\mathbf{C}_a(M1^{\otimes n}, M1^{\otimes m})$, we can canonically extend M to all $\mathcal{S}_\Sigma$ because each term formation rule (5)–(9) corresponds to an operation in graded SMCs.*

*A **model** of T valued in $\mathbf{C}$ is such an object $M1$ and assignment M satisfying $M(s) = M(t)$ for all axioms $s = t \in E_{\mathsf{h}}$.*

As for (ungraded) monoidal algebra in Section 2, we will see models as functors, so we need to define a syntactic category for a theory T. As usual, its morphisms will be terms quotiented by a congruence. In this case, the congruence is generated by the axioms in E_{h} and the necessary equations to satisfy the laws of a $\mathbb{G}$-graded SMC.

▶ **Definition 19** (Closure). *Given a theory $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$, we define the **closure** of E_{h}, denoted $\overline{E_{\mathsf{h}}}$, to be the smallest congruence on $\mathcal{S}_\Sigma$ (i.e. equivalence relation compatible with $\fatsemi$, $\otimes$, and $\star$) containing E_{h} and equations (G1)–(G11) in Figure 1, where terms are universally quantified.*

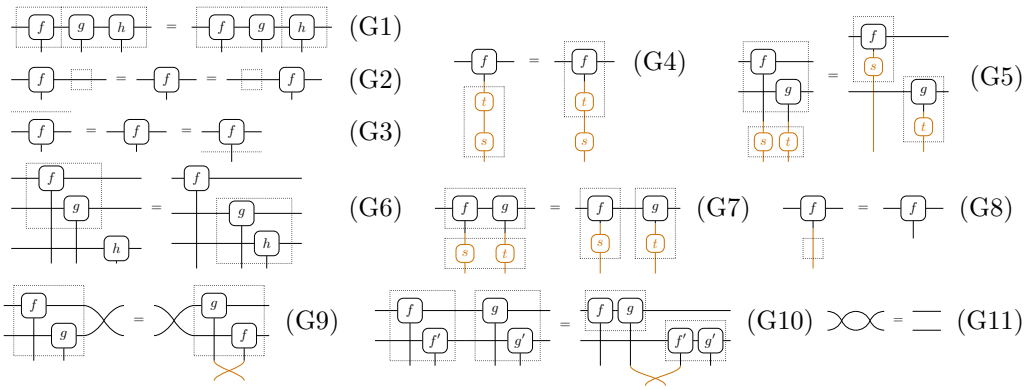
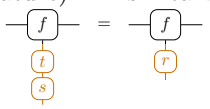


Figure 1 Laws of graded symmetric strict monoidal categories.

▶ **Definition 20** (Syntactic category). *The* ***syntactic category*** *of a $\mathbb{G}$-graded monoidal theory $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$ is a $\mathbb{G}$-graded prop $\mathcal{S}_{\mathsf{T}}$ defined as follows. The set of morphisms $\mathcal{S}_{\mathsf{T}a}(n, m)$ is the set of Σ-terms of arity n, coarity m, and grade a, quotiented by the congruence $\overline{E_{\mathsf{h}}}$. Identities, symmetries, sequential composition, monoidal product, and regrading are defined via the corresponding term formation rules (5)–(9). The axioms in Figure 1 ensure that all the laws of graded SMCs hold. We say that T is a* ***(graded) presentation*** *of a $\mathbb{G}$-graded prop $\mathbf{C}$ (or simply that T presents $\mathbf{C}$) if $\mathcal{S}_{\mathsf{T}} \cong \mathbf{C}$.*

▶ Remark 21. Strictly speaking, our notion of graded term is not purely syntactic, since the regrading rule (7) involves arbitrary $\mathbb{G}$-morphisms (which may or may not be freely obtained from a signature). This means that some equations are automatically imposed, e.g. if $s; t = r$ in $\mathbb{G}$, then  holds in $\mathcal{S}_{\mathsf{T}}$ by reflexivity. We provide this formulation for the sake of generality. However, if $\mathbb{G}$ was presented by a monoidal theory $(\Sigma_{\mathsf{v}}, E_{\mathsf{v}})$, we could have treated graded terms as pure syntax, by requiring that t appearing in the regrading rule (7) is a Σ_{v}-term rather than a morphism (= an equivalence class of Σ_{v}-terms). With this setup, we still obtain a definition of syntactic category for a graded theory equivalent to the one of Definition 20, by requiring that graded Σ-terms are further quotientied by the equations of E_{v} and the laws of SMCs (applied to the Σ_{v}-terms appearing in the Σ-terms as grading). In our leading example (Section 5.2), $\mathbb{G}$ is indeed obtained from a monoidal theory.

▶ **Theorem 22** (Soundness). *Models of T valued in $\mathbf{C}$ are in 1-to-1 correspondence with $\mathbb{G}$-graded symmetric monoidal functors $\mathcal{S}_{\mathsf{T}} \to \mathbf{C}$.*

Completeness is shown by considering the model of T that corresponds to $\mathrm{id} : \mathcal{S}_{\mathsf{T}} \to \mathcal{S}_{\mathsf{T}}$.

▶ **Theorem 23** (Completeness). *If $s = t$ is satisfied in all models of (Σ, E_{h}), then $s = t \in \overline{E_{\mathsf{h}}}$.*

The definitions and results of this section culminate into a conservative extension of (classical) monoidal theories and props. In fact, if we restrict Figure 1 to 0-graded terms, we recover the usual equations of SMCs. This requires the fact that 0 is terminal in $\mathbb{G}$, so all morphisms $0 \to 0$ are the identity. Moreover, if 0 is **strictly terminal** (i.e. there is no morphism $0 \to a$ other than $\mathrm{id}_0 : 0 \to 0$), then 0-graded terms in $\mathcal{S}_{\mathsf{T}}$ are precisely those built out of 0-graded generators. This implies that the underlying prop of a syntactic category $\mathcal{S}_{\mathsf{T}}$ is the syntactic category (Definition 5) for the monoidal theory given by the 0-graded generators and 0-graded equations in T.

▶ **Lemma 24.** *Suppose that* $\mathbb{G}(0,a) = \emptyset$ *whenever* $a \neq 0$*, and let* $\mathsf{T}_0 \coloneqq (\Sigma_0, E_0)$ *be the monoidal theory obtained by taking only the* 0*-graded generators and equations in a graded theory* $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$*. The underlying prop of* $\mathcal{S}_{\mathsf{T}}$ *is the syntactic category* $\mathcal{M}_{\mathsf{T}_0}$ *of* T_0*.*

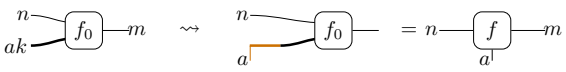
4 Modular Completeness for Graded Theories

In monoidal algebra it is typical to study a certain prop $\mathbf{C}$ of interest through the lenses of string diagrammatic calculi. Within this perspective, a key result is finding a monoidal theory T presenting $\mathbf{C}$ (in the sense of Definition 5). That would mean that we can reason about $\mathbf{C}$-morphisms entirely in algebraic terms, by manipulation of the string diagrams of $\mathcal{M}_{\mathsf{T}}$ via the equations of T.

In the case of a graded prop, there are two categories at stake: the base category $\mathbf{C}$, and the grading category $\mathbb{G}$. It is natural to ask the following question: can we systematically derive a presentation of the graded prop from a known presentation of both $\mathbf{C}$ and $\mathbb{G}$? In this section we outline conditions under which such question may be answered positively.

As above, we fix a semicartesian prop $\mathbb{G}$, and we assume it has a presentation $(\Sigma_{\mathsf{v}}, E_{\mathsf{v}})$. We further suppose that 0 is strictly terminal, i.e. there is no term  . We will show that, given a symmetric monoidal action $\bullet : \mathbb{G} \times \mathbf{C} \to \mathbf{C}$, a presentation of a prop $\mathbf{C}$ can be adapted into a graded presentation of $\mathbf{Para}(\mathbb{G}, \mathbf{C})$. Let us fix a monoidal presentation $(\Sigma_{\mathbf{C}}, E_{\mathbf{C}})$ of $\mathbf{C}$.

Since both $\mathbb{G}$ and $\mathbf{C}$ are props – in particular, their objects are either 0 or $n = (1+ \overset{n}{\cdots} +1)$ – the action $\bullet$ is completely determined on objects by the value of $1 \bullet 0$. Indeed, letting $k \coloneqq 1 \bullet 0$, we can use the properties of symmetric monoidal actions to prove that $a \bullet n = n + ak$ for all $a, n \in \mathbb{N}$. Now, a graded morphism $f : n \xrightarrow{a} m$ in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ corresponds, by definition, to a morphism $f_0 : a \bullet n \to m$ in $\mathbf{C}$. The presentation of $\mathbf{C}$ allows us to represent f_0 as a string diagram with $n + ak$ input wires and m output wires. To obtain a graded string diagram that represents f, we will simply bend down the bunch of ak input wires in f_0 into a grading wires as shown below.



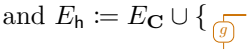
We make this visually intuitive process formal by defining a graded presentation of $\mathbf{Para}(\mathbb{G}, \mathbf{C})$.

The graded signature Σ contains all the generators in $\Sigma_{\mathbf{C}}$ seen as 0-graded generators, plus an additional 1-graded generator  $: 0 \xrightarrow{1} k$, where $k = 1 \bullet 0$. As explained before Lemma 24, there is a correspondence between monoidal $\Sigma_{\mathbf{C}}$-terms and 0-graded Σ-terms because we did not add 0-graded generators. Thus, all the axioms in $E_{\mathbf{C}}$ can be interpreted as equations between graded terms which we put in E_{h}. These will be the only equations between 0-graded terms so that 0-graded string diagrams correspond to morphisms in $\mathbf{C}$.

The remaining equations in E_{h} will reflect the definition of regrading in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ (cf. (15) in Section B). For each vertical generator  in the presentation of $\mathbb{G}$, we add the equation on the left in (10), where g_0 is a $\Sigma_{\mathbf{C}}$-term in the equivalence class of the morphism $g \bullet \mathrm{id}_0 \in \mathbf{C}$, seen as a 0-graded Σ-term. We also add the same equation for g being the symmetry which is drawn on the right in (10), using that $\sigma_{1,1} \bullet \mathrm{id}_0 = \sigma_{k,k}$.

(10)

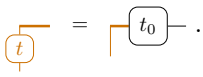
With slight abuse of notation we write  for the generator and the a-wise monoidal product of with itself for $a \in \mathbb{N}$ (the product order is inconsequential by (G6)). We use thick wires to emphasize that there may be a different number of grading wires and output wires in . Concisely, $\mathsf{T} \coloneqq (\Sigma, E_{\mathsf{h}})$, where

$$\Sigma \coloneqq \Sigma_{\mathbf{C}} \sqcup \{ \text{} : 0 \xrightarrow{1} 1 \} \text{ and } E_{\mathsf{h}} \coloneqq E_{\mathbf{C}} \cup \{ \text{  } \mid \text{  } \in \Sigma_{\mathsf{v}} \cup \{ \text{} \} \}. \tag{11}$$

It is important to note that, by construction, the underlying prop of $\mathcal{S}_{\mathsf{T}}$ is isomorphic to $\mathcal{M}_{\Sigma_{\mathbf{C}},E_{\mathbf{C}}}$, and hence to $\mathbf{C}$. Moreover, since the underlying prop of $\mathbf{Para}(\mathbb{G},\mathbf{C})$ is $\mathbf{C}$, this establishes an isomorphism between the underlying props of $\mathcal{S}_{\mathsf{T}}$ and $\mathbf{Para}(\mathbb{G},\mathbf{C})$. It sees a 0-graded string diagram as a diagram in $\mathcal{M}_{\Sigma_{\mathbf{C}},E_{\mathbf{C}}}$, then sends it to the corresponding morphism in $\mathbf{C}$ through the presentation, and finally to that same morphism seen as a 0-graded morphism of $\mathbf{Para}(\mathbb{G},\mathbf{C})$. Our goal is now to make this an isomorphism between the whole graded props.

Our argument relies on finding a particular shape that terms can be put into (e.g. a normal form). Here, we want to factor any graded term as the composition of a very simple graded term and a $\Sigma_{\mathbf{C}}$-term, showing that any graded term is obtained by bending wires of a 0-graded term. We need a technical lemma that generalises (10) to all $\mathbb{G}$-morphisms.

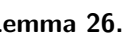
▶ **Lemma 25.** *For any morphism* t *in* $\mathbb{G}$*, and any monoidal* $\Sigma_{\mathbf{C}}$*-term* t_0 *in the equivalence class of* $t \bullet \mathrm{id}_0 \in \mathbf{C}$*, the closure* $\overline{E_{\mathsf{h}}}$ *contains the equation*



(12)

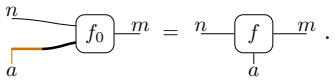
Proof. Since $\mathbb{G}$ is presented by a monoidal theory $(\Sigma_{\mathsf{v}}, E_{\mathsf{v}})$, we can proceed by induction on a representative monoidal Σ_{v}-term of t. The base cases (when t is a generator or a symmetry) are handled by the axioms that are in E_{h} (10). The inductive cases follow by diagrammatic reasoning after noting that $(s;t)_0 = s_0;t_0$ and $(s \otimes t)_0 = s_0 \otimes t_0$ are provable in T. ◀

We can now factor any term. One may visualise this process as sliding all the $\mathbb{G}$-morphisms through $\ulcorner$, then pulling back the vertical dangling wires all the way to the left.

▶ **Lemma 26.** *Given a graded* Σ*-term* 

, there is a term 

such that



(13)

Proof. We construct f_0 explicitly by induction on f. The base cases are trivial, and the inductive cases follow from diagrammatic reasoning. ◀

▶ **Theorem 27.** *Let* $\mathbf{C}$ *be a symmetric monoidal* $\mathbb{G}$*-actegory and* $(\Sigma_{\mathbf{C}}, E_{\mathbf{C}})$ *a presentation of the SMC* $\mathbf{C}$*. The theory* $\mathsf{T} = (\Sigma, E_{\mathsf{h}})$ *defined in* (11) *is a graded presentation of* $\mathbf{Para}(\mathbb{G},\mathbf{C})$*.*

Proof. We need to construct an isomorphism $[\![-]\!] : \mathcal{S}_{\mathsf{T}} \to \mathbf{Para}(\mathbb{G},\mathbf{C})$.

We will use Theorem 22 to inductively define $[\![-]\!]$ by giving its value on generators. Following the discussion above, $[\![-]\!]$ sends each 0-graded generator to the corresponding 0-graded morphism in $\mathbf{Para}(\mathbb{G},\mathbf{C})$ via $\mathcal{S}_{\mathsf{T}0}(n,m) \cong \mathcal{M}_{\Sigma_{\mathbf{C}},E_{\mathbf{C}}}(n,m) \cong \mathbf{C}(n,m) \cong \mathbf{Para}(\mathbb{G},\mathbf{C})_0(n,m)$. For the remaining generator $\ulcorner : 0 \xrightarrow{1} 1$, we define $[\![\ulcorner]\!]$ to be the morphism resulting from the action of id_1 in $\mathbb{G}$ on id_0 in $\mathbf{C}$, namely, $[\![\ulcorner]\!] := \mathrm{id}_1 \bullet \mathrm{id}_0 : 0 \xrightarrow{1} 1$.

The assignment $[\![-]\!]$ is canonically extended to graded Σ-terms, and it is easy to see that it acts exactly like the isomorphism $\mathcal{M}_{\Sigma_{\mathbf{C}},E_{\mathbf{C}}} \to \mathbf{C}$ on 0-graded terms. Therefore, for all axioms $f = g \in E_{\mathbf{C}}$, we have $[\![f]\!] = [\![g]\!]$. For the remaining axioms in E_{h}, those coming from (10), we have for any morphism $t : b \to a \in \mathbb{G}$,

$$\left[\!\!\left[\begin{array}{c} t \\ \ulcorner \end{array} \right]\!\!\right] = t \star (\mathrm{id}_a \bullet \mathrm{id}_0) = (\mathrm{id}_b \bullet \mathrm{id}_0) \fatsemi (t \bullet \mathrm{id}_0) = \left[\!\!\left[\ulcorner \; t_0 \right]\!\!\right].$$

We have shown that all axioms in E_{h} hold in $\mathbf{Para}(\mathbb{G},\mathbf{C})$, thus, we defined a model of T, and by Theorem 22, $[\![-]\!]$ factors through an identity-on-objects functor $[\![-]\!] : \mathcal{S}_{\mathsf{T}} \to \mathbf{Para}(\mathbb{G},\mathbf{C})$.

To show that $[\![-]\!]$ is in fact an isomorphism, it suffices to prove it is fully faithful. This relies on the fact that $[\![-]\!]$ is an isomorphism (hence fully faithful) when restricted to 0-graded terms, and the decomposition in Lemma 26. ◀

5 Case Study: Imprecise Probability

In this section, we use the previously introduced framework (in particular, Theorem 27) to completely axiomatise a graded category **BImP** of imprecise probabilistic processes.

5.1 BinStoch: Binary Stochastic Matrices

Let us first recall the definition of **BinStoch** and its presentation in [42].

▶ **Definition 28** (**BinStoch**). *An $m \times n$ **stochastic matrix** is a matrix with m rows and n columns with entries in the interval $[0,1]$ such that, in each column, the sum of all the entries is 1. The symmetric monoidal category* **FinStoch** *has objects the natural numbers and as morphisms $n \to m$ the $m \times n$ stochastic matrices, with sequential composition given by matrix multiplication. On objects, $n \otimes m$ is the usual product of integers. On matrices, given $A : n \rightsquigarrow n'$ and $A' : m \rightsquigarrow m'$, $A \otimes A' : nn' \to mm'$ is defined in block matrix form as the Kronecker product $A \otimes A' := \begin{bmatrix} a_{11}A' & \dots & a_{1n}A' \\ \vdots & \ddots & \vdots \\ a_{n'1}A' & \dots & a_{n'n}A' \end{bmatrix}$.*

The category **BinStoch** *has objects the natural numbers, and as morphisms $n \to m$ the $2^m \times 2^n$ stochastic matrices:* $\mathbf{BinStoch}(n,m) = \mathbf{FinStoch}(2^n, 2^m)$. *All the structure of* **BinStoch** *is obtained by seeing it as a full subcategory of* **FinStoch** *with the inclusion* **BinStoch** $\hookrightarrow$ **FinStoch** *sending n to 2^n. In particular, the monoidal product on objects is addition since $2^n \otimes 2^m = 2^n 2^m = 2^{n+m}$, thus* **BinStoch** *is a prop. There is always a unique morphism $n \to 0$, the matrix with a single row filled with 2^n ones, so 0 is terminal. Moreover, 0 is the unit of the monoidal product as $0 + n = n$, hence* **BinStoch** *is semicartesian.*

In short, one may regard **BinStoch** as a "binary" version of **FinStoch**. It is interesting for our developments because, unlike **FinStoch**, a monoidal theory presenting **BinStoch** is known[1]. We recall such theory from [42], as we will leverage it for our axiomatisation result.

▶ **Definition 29** ([42]). *The monoidal theory T_{cc} has generators*  ,  ,  ,  , *and*  *for all $p \in [0,1]$, and equations as in [42, Figure 4]. Following [42], we write* CausCirc *for the syntactic category $\mathcal{S}_{\mathsf{T}_{\mathsf{cc}}}$.*

▶ **Theorem 30** ([42], Corollary 3.18). *T_{cc} presents* **BinStoch**.

It is worth providing some intuition on how the isomorphism $[\![-]\!] : \mathsf{CausCirc} \cong \mathbf{BinStoch}$ works. We may regard $[\![-]\!]$ as a semantics, giving an operational meaning to string diagrams of CausCirc as circuits. First, a single wire  (the identity $1 \to 1$) carries **probabilistic bits**, that is, convex combinations $p|\mathtt{1}\rangle + (1-p)|\mathtt{0}\rangle = \begin{bmatrix} p \\ 1-p \end{bmatrix}$. We interpret a probabilistic bit $p|\mathtt{1}\rangle + (1-p)|\mathtt{0}\rangle$ as a probability distribution φ on $\mathbb{B} = \{\mathtt{1}, \mathtt{0}\}$ with $\varphi(\mathtt{1}) = p$ and $\varphi(\mathtt{0}) = 1-p$. The semantics of the identity wire is the identity 2×2 matrix: it leaves the bit unchanged.

A double wire  carries two probabilistic bits. However, probabilistic bits can be *correlated*, not unlike entangled qubits in quantum circuits. Formally, a double wire carries distributions on $\mathbb{B} \otimes \mathbb{B} := \{\mathtt{11}, \mathtt{10}, \mathtt{01}, \mathtt{00}\}$. If φ and ψ are two probabilistic bits, then their Kronecker product is a distribution on $\mathbb{B} \otimes \mathbb{B}$. However, a distribution on $\mathbb{B} \otimes \mathbb{B}$ does not always arise as the independent product of two probabilistic bits.

[1] To be precise, a presentation of **FinStoch** is known when the monoidal product is given by direct sum of matrices [22], but not with the Kronecker product. This difference is highly relevant in applications, because parallel probabilistic processes may be correlated only with the latter.

Any morphism in $\mathbf{BinStoch}(0, n)$ is a **state** that "emits" n probabilistic bits, and it corresponds to a column vector of dimension 2^n whose entries sum up to 1 – the entry in the ith row will be the weight of the distribution at $|\mathtt{bin}(2^n - i)\rangle$, where $\mathtt{bin}(2^n - i)$ is the binary representation of $2^n - i$. For $n = 1$, and given $p \in [0, 1]$, the generator  emits $p|\mathtt{1}\rangle + (1-p)|\mathtt{0}\rangle = \left[\begin{smallmatrix} p \\ 1-p \end{smallmatrix}\right]$. In particular,  emits $|\mathtt{1}\rangle$ and emits $|\mathtt{0}\rangle$. The generator  behaves as **discard**: it accepts any bit coming in and outputs nothing. The generator  behaves as **copy**: it takes an input $p|\mathtt{1}\rangle + (1-p)|\mathtt{0}\rangle$ and creates two entangled bits $p|\mathtt{11}\rangle + (1-p)|\mathtt{00}\rangle$. The remaining generators  and are **and** and **not**: they behave like their classical circuits counterpart sending $|x\rangle|y\rangle$ to $|x \ \mathtt{and} \ y\rangle$ and $|x\rangle$ to $|\mathtt{not}\ x\rangle$ respectively.

5.2 BImP: Binary Imprecise Probability

The recent work [34] studies a graded category **ImP** as a model for probabilistic programming languages based on imprecise probability. Formally, **ImP** is defined as the graded category resulting from the **Para** construction $\mathbf{Para}(\mathbf{FinStoch}_{\mathrm{Surj}}, \mathbf{FinStoch})$ (*cf.* Example 13). That is, **ImP**-morphisms are stochastic matrices with grading from the **FinStoch**-subcategory of surjective stochastic matrices. Intuitively, the grading indicates the nondeterministic choices of a probabilistic process.

Our goal is to identify the monoidal theory presenting a variant of **ImP**. The reason for not tackling **ImP** directly is that, as mentioned in Section 5.1, a presentation of **FinStoch** is not known. However, as we will see in Section 6, our variant suffices to provide a semantic model for imprecise probabilistic programming. We tweak **ImP** along two axes. First, for the base category, we pick **BinStoch**, the binary version of **FinStoch**, which unlike its sup-category enjoys a monoidal presentation (Theorem 30). Second, we also restrict the grading category. As remarked in [34], $\mathbf{FinStoch}_{\mathrm{Surj}}$ is an "overly generous" choice as grading category: for their developments, it suffices to consider only the (deterministic) surjections $2^B \to 2^A$ that are images of injections $A \to B$ under the contravariant powerset functor $2^- : \mathbf{Set} \to \mathbf{Set}$. The prop corresponding to this class of morphisms is $\mathbf{FinInj}^{\mathrm{op}}$, the opposite of the skeleton category of finite sets and injections. In other words, $\mathbf{FinInj}^{\mathrm{op}}$ has objects the natural numbers and a morphism $n \to m$ is an injection $\underline{m} \to \underline{n}$, where $\underline{n} \coloneqq \{0, \ldots, n-1\}$.

Just as $\mathbf{FinStoch}_{\mathrm{Surj}}$ is a subcategory of **FinStoch**, we can view $\mathbf{FinInj}^{\mathrm{op}}$ as a subprop of **BinStoch** with the functor $I : \mathbf{FinInj}^{\mathrm{op}} \hookrightarrow \mathbf{BinStoch}$ defined as follows. It is an identity-on-objects functor $I(n) = n$, and it sends an injective function $f : \underline{m} \to \underline{n}$ (a morphism $n \to m$ in $\mathbf{FinInj}^{\mathrm{op}}$) to the stochastic matrix $I(f)$ that sends $|\mathtt{x}_0 \cdots \mathtt{x}_{n-1}\rangle$ to $|\mathtt{x}_{f(0)} \cdots \mathtt{x}_{f(m-1)}\rangle$. For example, if $f : \underline{2} \to \underline{2}$ is the function that swaps 0 and 1, then $I(f) = \sigma_{1,1} =$  .

We may now obtain a symmetric monoidal $\mathbf{FinInj}^{\mathrm{op}}$-actegory $\bullet : \mathbf{FinInj}^{\mathrm{op}} \times \mathbf{BinStoch} \to \mathbf{BinStoch}$ defined on objects by $(n, m) \mapsto n + m$, and on morphisms by $(f, A) \mapsto I(f) \otimes A$. Applying the Para construction (Example 13), we obtain $\mathbf{Para}(\mathbf{FinInj}^{\mathrm{op}}, \mathbf{BinStoch})$, which we call **BImP**. Just as **BinStoch** compared to **FinStoch**, we regard **BImP** as a binary version of **ImP**. We now make explicit the structure of **BImP**. Its objects are natural numbers. For any $n, m, a \in \mathbb{N}$, the set of morphisms $n \to m$ with grade a is $\mathbf{BImP}_a(n, m) \coloneqq \mathbf{BinStoch}(a \otimes n, m) = \mathbf{FinStoch}(2^{a+n}, 2^m)$. Regrading, composition, and monoidal product in **BImP** are defined as follows (occurrences of $\otimes$ on the right-hand side are in **BinStoch**):

$$r \star f \coloneqq b \otimes n \xrightarrow{I(r) \otimes \mathrm{id}_n} a \otimes n \xrightarrow{f} m$$

$$f \fatsemi g \coloneqq (a \otimes b) \otimes n \xrightarrow{\sigma_{a,b} \otimes \mathrm{id}_n} b \otimes a \otimes n \xrightarrow{\mathrm{id}_b \otimes f} b \otimes m \xrightarrow{g} \ell$$

$$f \otimes f' \coloneqq (a \otimes b) \otimes n \otimes n' \xrightarrow{\mathrm{id}_a \otimes \sigma_{b,n} \otimes \mathrm{id}_{n'}} a \otimes n \otimes b \otimes n' \xrightarrow{f \otimes f'} m \otimes m'.$$

Now, how are these matrices interpreted as imprecise probabilistic processes? A morphism $f \in \mathbf{BImP}_a(n,m)$ corresponds to an $2^m \times 2^a 2^n$ stochastic matrix. We may regard f as decomposed into 2^a stochastic **submatrices** $f|\vec{x}\rangle\!\langle -|$ of dimension $2^m \times 2^n$, with $\vec{x}$ ranging in $\mathbb{B}^a$, where $f|\vec{x}\rangle\!\langle -|$ denotes the morphism in **BinStoch** sending $|\vec{u}\rangle$ to $f|\vec{x}\rangle\!\langle\vec{u}|$. To unravel f, we write all the submatrices in decreasing order of the number represented by $\vec{x}$ and separate them with vertical lines. For example, $f = [\, f|1\rangle\!\langle -| \mid f|0\rangle\!\langle -| \,] = \left[\begin{smallmatrix} 1 & 0.5 \\ 0 & 0.5 \end{smallmatrix}\right]$ represents a morphism $f : 0 \xrightarrow{1} 1$ in **BImP**. It is decomposed as $2^1 = 2$ stochastic matrices of dimension $2^1 \times 2^0$ separated by a vertical line. Similarly to the interpretation of **ImP** given in [34], we view the separations as nondeterministic choices between the matrices on either side. In the example of $f = \left[\begin{smallmatrix} 1 & 0.5 \\ 0 & 0.5 \end{smallmatrix}\right]$, each choice is a column vector, thus a distribution. Compared with the algebraic syntax used in e.g. [11], f may also be written as $\delta_1 \oplus (\delta_1 +_{0.5} \delta_0)$.[2]

5.3 Graded Presentation of BImP

We now have all the ingredients to give an axiomatisation of **BImP**. First, it is well known that the prop $\mathbf{FinInj}^{\mathrm{op}}$ is presented by a monoidal theory containing a single generator and no equations, see e.g. [53, Example 2.28(a)]. As discussed, the prop **BinStoch** is presented by the monoidal theory T_{cc} (Theorem 30). The graded prop **BImP** is defined as $\mathbf{Para}(\mathbf{FinInj}^{\mathrm{op}}, \mathbf{BinStoch})$. Therefore, by Theorem 27, we obtain modularly a presentation of **BImP** by regarding T_{cc} as a 0-graded theory, and adding a generator with the equations (10) for the generator $g =$ and the symmetry $g =$. There are many choices for the terms representing g_0, but the most natural are the discard generator in T_{cc} and the swapping of wires, thus the additional axioms are (no need for thick wires since $1 \bullet 0$ in this setting)

(14)

▶ **Theorem 31.** *The graded theory obtained by Theorem 27 for* $\mathbf{Para}(\mathbf{FinInj}^{\mathrm{op}}, \mathbf{BinStoch})$ *described above, denote it* $\mathsf{T}_{\mathsf{cc}_i}$ *is a graded presentation of* **BImP**.

We write $\mathsf{CausCirc}_i$ (imprecise $\mathsf{CausCirc}$) for the syntactic category of $\mathsf{T}_{\mathsf{cc}_i}$. We can read string diagrams of $\mathsf{CausCirc}_i$ with the same interpretation as those of $\mathsf{CausCirc}$, except we view the vertical wires as carrying nondeterministic bits carrying one of $|1\rangle$ or $|0\rangle$.

▶ Remark 32. We can now reveal how the choice of putting the grading wire on the bottom makes more visual sense. In Lemma 25, we prove a generalisation of the axiom (10) for all $\mathbb{G}$-morphisms. In the theory presenting **BImP**, this result becomes visually intuitive. Any monoidal term representing a morphism in $\mathbf{FinInj}^{\mathrm{op}}$ can slide through the generator (or a-wise monoidal products of it) after a doing a 90 degree rotation as drawn below. If the grading wires were coming from the top, this generalised axiom would involve a reflection and a rotation. This is simply an artefact of the choice of reading convention.

[2] δ_i denotes the Dirac distribution on $|i\rangle$. There is a caveat because $\oplus$ models a commutative nondeterministic choice in [11], while in **BImP**, $\left[\begin{smallmatrix} 1 & 0.5 \\ 0 & 0.5 \end{smallmatrix}\right] \neq \left[\begin{smallmatrix} 0.5 & 1 \\ 0.5 & 0 \end{smallmatrix}\right]$. This is further discussed in [34].

6 Imprecise Probabilistic Programming

The primary goal of [34] is to provide theoretical foundations for a compositional language modelling imprecise probability. They prove that their framework of choice, the category **ImP**, is a convenient setting to interpret such a language. The category **BImP** has similar features, but it also reaps the benefits of the string diagrammatic presentation we gave in Theorem 31. In particular, it allows for a graphical translation of programs. In Section 6.1, we define a programming language with nondeterministic and probabilistic choices that can be translated into string diagrams of **BImP**. We also show that this language is commutative and affine, i.e. the equations in Theorem 33 hold, hence it satisfies *Desideratum 1* of [34]. In Section 6.2, we extend the language with a construct for exact conditioning by "augmenting" **BImP** mirroring how CausCirc is augmented to ProbCirc in [42]. This is immediate thanks to Theorem 27. We end the section by presenting two examples of graphical programs that illustrate the potential in modelling imprecise probabilities.

6.1 Programming in BImP

We extend the language studied in [42] (without `observe` for now) with the `knight` construct for nondeterministic choices discussed in [34]. We inductively define the syntax:

$$
\begin{aligned}
e, e_1, e_2 ::= {} & x \mid \texttt{flip}(p) \mid \texttt{knight} \mid \langle e_1, e_2 \rangle \mid \texttt{fst}\ e \mid \texttt{snd}\ e \mid \\
& \ldots \mid \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 && \text{(expressions)} \\
\tau, \tau_1, \tau_2 ::= {} & \mathbb{B} \mid \tau_1 \otimes \tau_2 && \text{(types)} \\
\Gamma ::= {} & \emptyset \mid \Gamma, x : \tau, && \text{(contexts)}
\end{aligned}
$$

where x ranges over a countable set of variables, p ranges over $[0, 1]$, and contexts are considered modulo contraction and exchange (i.e. as sets of typed variables). The intended semantics for this language is rather natural: $\texttt{flip}(p)$ generates a probabilistic bit $p|\texttt{1}\rangle + (1-p)|\texttt{0}\rangle$, `knight` generates a nondeterministic bit that is $|\texttt{1}\rangle$ or $|\texttt{0}\rangle$, the pairing, projections, and `if then else` have their usual semantics, and similarly for the `let` primitive, but we note that the latter is essential to allow reusing the result of a process (*cf.* named choices in [34]).

The type and grade of an expression can be inferred with the rules in Figure D.1 (*cf.* [42, Fig. 2] and [34, Sec. 2.3]), where $\Gamma \vdash e : \tau \,\&\, a$ should be read as "e has type τ and grade a in the context Γ". We interpret types and contexts as objects in **BImP** inductively: $[\![\mathbb{B}]\!] = 1$; $[\![\tau_1 \otimes \tau_2]\!] = [\![\tau_1]\!] + [\![\tau_2]\!]$; $[\![\emptyset]\!] = 0$; $[\![\Gamma, x : \tau]\!] = [\![\Gamma]\!] + [\![\tau]\!]$.A typable expression $\Gamma \vdash e : \tau \,\&\, a$ is interpreted as a morphism $[\![\Gamma]\!] \xrightarrow{a} [\![\tau]\!]$ in **BImP**. By Theorem 31, we can represent such morphisms with graded string diagrams built out of the generators of CausCirc and ⌜. In Figure 2, we define the semantics of typable expressions as such diagrams by induction on the typing derivation (*cf.* [42, Fig. 3]).

We can show using diagrammatic reasoning that our language is commutative and affine.

▶ **Theorem 33.** *The programs below (within appropriate contexts) are equal up to regrading. For commutativity and weakening, we suppose that u does not use x.*

$$
\begin{aligned}
\texttt{let}\ x = t\ \texttt{in}\ (\texttt{let}\ y = u\ \texttt{in}\ v) &\approx \texttt{let}\ y = (\texttt{let}\ x = t\ \texttt{in}\ u)\ \texttt{in}\ v && \textit{(associativity)} \\
\texttt{let}\ x = t\ \texttt{in}\ (\texttt{let}\ y = u\ \texttt{in}\ v) &\approx \texttt{let}\ y = u\ \texttt{in}\ (\texttt{let}\ x = t\ \texttt{in}\ v) && \textit{(commutativity)} \\
\texttt{let}\ x = t\ \texttt{in}\ u &\approx u && \textit{(weakening)}
\end{aligned}
$$

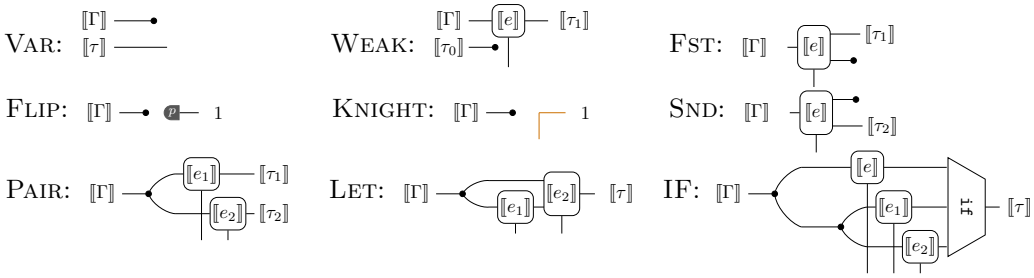
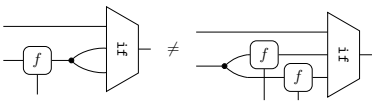


Figure 2 Inductive definition of the semantics of our language. The trapezium node labelled `if` is syntactic sugar for a diagram of type $1 + n + n \xrightarrow{0} n$ for any $n \in \mathbb{N}$. Intuitively, it reads the input of the first wire and when it is true, it outputs the inputs of the next n wires, and when it is false, it outputs the inputs of the last n wires. It is defined as the multiplexer in [42, Examples 2.1 and 2.2].

▶ Remark 34. The hoisting equation that corresponds to *Desideratum 2* in [34] does not hold in our semantics because the following two diagrams are not equal in general.



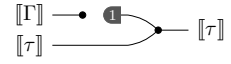
Indeed, the grades of both diagrams do not match ($a + a \neq a$ unless $a = 0$). What is more, the copy generator is not natural, meaning that not all boxes can be copied. Boxes that can be copied are called **deterministic** in the literature on Markov categories [23]. In **BImP**, a morphism is deterministic if it does not contain the generators $\overset{p}{\bullet}$ with $p \neq 0, 1$ nor $\ulcorner$. Abstractly, the reason for the failure of hoisting is that **BImP** does not support the full structure of a distributive graded Markov category that **ImP** has because, while $\mathbf{BImP}_0 = \mathbf{BinStoch}$ is a Markov category, there is no coproduct. We mention a potential solution in the conclusion.

6.2 Adding Exact Conditioning

The authors of [42] provide another presentation result that builds upon Theorem 30: they define a prop ProbCirc by adding a **conditioning** generator $\supset\!\bullet$ and equations to CausCirc, then they show that ProbCirc is isomorphic to the prop of substochastic matrices [42, Theorem 4.3]. The semantics of $\supset\!\bullet$ is a substochastic map that sends $|x\rangle|y\rangle$ to $|x\rangle$ if $x = y$ and has no output otherwise (i.e a distribution with total weight 0). Now, $\mathbf{FinInj}^{\mathrm{op}}$ also embeds inside ProbCirc (exactly like it does in CausCirc $\cong$ **BinStoch**), thus we can apply Theorem 27 to get an analogue of Theorem 31. Namely, we have a graded presentation of the substochastic variant of **BImP**. We do not go over the technical details, which differ very little from the presentation of **BImP**.

When programming in this new graded category, the `observe` primitive is available. It obeys the following typing rule and diagrammatic translation.

$$\Gamma, x : \tau \vdash \texttt{observe}\ x : \tau \,\&\, 0 \quad \rightsquigarrow$$



According to the semantics of the conditioning generator, `observe` x puts a condition on the program to require that x is true (i.e. equal to a product of $|1\rangle$s). This is an essential component of probabilistic programs. With this functionality, we can showcase two examples that underscore the role of nondeterminism in imprecise probability.

Figure 3 Graphical representation of two versions of the Boy or Girl paradox.

▶ **Example 35** (Boy or Girl)**.** The *Boy or Girl paradox* [51] is a well-known probabilistic riddle that we can state as follows.

> Your neighbours have two kids and at least one of them is a girl. Assuming that the sex of each kid is assigned with a fair coin flip (50% chance of boy and 50% chance of girl) independently of each other, what is the probability that both kids are girls?

We model this riddle with a probabilistic program, which we immediately translate to a diagram in **BImP** (Figure 3 on the left), where ⟩⊃− is defined in [42] and it sends $|x\rangle|y\rangle$ to $|x \texttt{ or } y\rangle$. This conveys an operational interpretation of the riddle. First, the sex of both kids is stored in two probabilistic bits determined by two independent coin flips represented with the generator 1/2— . Then, in parallel, we require the `or` of these bits to be `1`, meaning that at least one bit is a `1` (read *a girl*), and we output the bits. Applying the functor $[\![-]\!]$ yields a 1×4 matrix corresponding to the subdistribution $1/4|\texttt{11}\rangle + 1/4|\texttt{10}\rangle + 1/4|\texttt{01}\rangle$. To answer the riddle, we take the probability of having two girls relative to the total probability: $\frac{1/4}{3/4} = 1/3$.

A variant of this riddle can be stated as follows.

> Your neighbours have two kids: a toddler and a teen. You meet one of them, and she is a girl. What is the probability that both kids are girls?

In this version, there are two possible events, either you learn that the toddler is a girl, or you learn that the teen is a girl. Both events give you more information than in the original riddle. This riddle is modelled in Figure 3 on the right. As before, two bits represent the sex of each kid. We condition on the first bit (say the toddler) being a `1` and, in parallel, we condition on the second bit being a `1`. We pick one of these two possibilities with an `if` block whose *guard* wire takes a nondeterministic bit. Applying the functor $[\![-]\!]$ yields two 1×4 submatrices corresponding to the subdistributions $1/4|\texttt{11}\rangle + 1/4|\texttt{10}\rangle$ and $1/4|\texttt{11}\rangle + 1/4|\texttt{01}\rangle$. In both cases, the relative probability of the neighbours having two girls is $1/2$.

▶ Remark 36. Adding the `observe` construct to our language breaks affineness, namely, the weakening property of Theorem 33 does not hold any more. Indeed, our proof of that property uses the naturality property (f •− = −•) which does not hold for the new generator $f = \texttt{observe}$. Concretely, the equation observe •− = =: cannot be true because if you condition on two bits being equal before discarding them, you are still putting some constraints on the rest of the process – that it led to those two bits being equal. The other two properties in Theorem 33, associativity and commutativity, still hold.

7 Conclusion

In this work, we laid the foundations of graded monoidal algebra, by introducing suitable notions of theory, syntactic prop generated by a theory, model, and string diagram. We proved a modularity result, which allows to reduce the task of axiomatising a graded prop to that of axiomatising the base category and the grading category. As case study, we axiomatised a graded prop whose morphisms represent imprecise probabilistic processes. We showed how a simple probabilistic programming language with nondeterminism may be interpreted in this category, and how this model may be extended with exact conditioning.

Our work was motivated by the study of imprecise probability through graded distributive Markov categories, as introduced by [34]. The diagrammatic language we introduced does not capture the full structure exploited in [34]. In particular, graded string diagrams do not naturally accomodate coproducts. This could be achieved by combining our approach with graphical languages for distributive categories [15, 6], leading to a theory of graded distributive string diagrams.

We plan to consider other instances of our framework. First, the axiomatisation of the category **Gauss** of Gaussian stochastic processes in [49] could be combined with grading wires that model nondeterminism, provided we describe the appropriate action of $\mathbf{FinInj}^{\mathrm{op}}$ on **Gauss**. Second, it is natural to ask whether our modular approach to axiomatisation may be applied to categories of algebras for graded (co)monads, as studied in programming language semantics [30, 24, 18, 25, 19, 21]. Finally, we would like to study grading of string diagrammatic calculi for automata and other state-based systems, as introduced in [43], drawing inspiration from the use of graded monads in trace semantics [39].

On a more abstract level, we are also interested in characterising the relation between our graded monoidal theories and graded algebraic theories (see e.g. [32, 47]) as the authors of [10] have done in the classical setting. This would allow obtaining graded monoidal presentations starting from graded algebraic presentations [31].

References

1 Thibaut Antoine, Robin Piedeleu, Alexandra Silva, and Fabio Zanasi. A Complete Diagrammatic Calculus for Automata Simulation. In Jörg Endrullis and Sylvain Schmitz, editors, *33rd EACSL Annual Conference on Computer Science Logic (CSL 2025)*, volume 326 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:22, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2025.27.

2 Miriam Backens, Aleks Kissinger, Hector Miller-Bakewell, John van de Wetering, and Sal Wolffs. Completeness of the ZH-calculus. *Compositionality*, Volume 5 (2023), July 2023. doi:10.32408/compositionality-5-5.

3 John C. Baez, Brandon Coya, and Franciscus Rebro. Props in network theory. *Theory Appl. Categ.*, 33:Paper No. 25, 727–783, 2018.

4 Guillaume Boisseau and Robin Piedeleu. Graphical piecewise-linear algebra. In *Foundations of software science and computation structures*, volume 13242 of *Lecture Notes in Comput. Sci.*, pages 101–119. Springer, Cham, 2022. doi:10.1007/978-3-030-99253-8_6.

5 Filippo Bonchi, Alessandro Di Giorgio, Nathan Haydon, and Pawel Sobocinski. Diagrammatic algebra of first order logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '24, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3661814.3662078.

6 Filippo Bonchi, Alessandro Di Giorgio, and Alessio Santamaria. Deconstructing the calculus of relations with tape diagrams. *Proc. ACM Program. Lang.*, 7(POPL), January 2023. doi:10.1145/3571257.

7 Filippo Bonchi, Alessandro Di Giorgio, and Elena Di Lavore. A diagrammatic algebra for program logics, 2024. doi:10.48550/arXiv.2410.03561.

8 Filippo Bonchi, Joshua Holland, Robin Piedeleu, Paweł Sobociński, and Fabio Zanasi. Diagrammatic algebra: from linear to concurrent systems. *Proc. ACM Program. Lang.*, 3(POPL), January 2019. doi:10.1145/3290338.

9 Filippo Bonchi, Dusko Pavlovic, and Pawel Sobocinski. Functorial semantics for relational theories. *CoRR*, abs/1711.08699, 2017. arXiv:1711.08699.

10 Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. Deconstructing Lawvere with distributive laws. *J. Log. Algebr. Methods Program.*, 95:128–146, 2018. doi:10.1016/j.jlamp.2017.12.002.

11 Filippo Bonchi, Ana Sokolova, and Valeria Vignudelli. The theory of traces for systems with nondeterminism, probability, and termination. *Logical Methods in Computer Science*, Volume 18, Issue 2, June 2022. doi:10.46298/lmcs-18(2:21)2022.

12 Silvio Capobianco and Tarmo Uustalu. Additive cellular automata graded-monadically. In *Proceedings of the 25th International Symposium on Principles and Practice of Declarative Programming*, PPDP '23, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3610612.3610625.

13 Matteo Capucci and Bruno Gavranović. Actegories for the working amthematician, March 2022. arXiv:2203.16351v1.

14 Matteo Capucci, Bruno Gavranović, Jules Hedges, and Eigil Fjeldgren Rischel. Towards foundations of categorical cybernetics. *Electronic Proceedings in Theoretical Computer Science*, 372:235–248, November 2022. doi:10.4204/eptcs.372.17.

15 Cole Comfort, Antonin Delpeuch, and Jules Hedges. Sheet diagrams for bimonoidal categories, 2020. arXiv:2010.13361.

16 Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. In *Programming languages and systems. 31st European symposium on programming, ESOP 2022, held as part of the European joint conferences on theory and practice of software, ETAPS 2022, Munich, Germany, April 2–7, 2022. Proceedings*, pages 1–28. Cham: Springer, 2022. doi:10.1007/978-3-030-99336-8_1.

17 Ivan Di Liberti, Fosco Loregian, Chad Nester, and Paweł Sobociński. Functorial semantics for partial theories. *Proc. ACM Program. Lang.*, 5(POPL), January 2021. doi:10.1145/3434338.

18 Ulrich Dorsch, Stefan Milius, and Lutz Schröder. Graded monads and graded logics for the linear time–branching time spectrum. In *30th International Conference on Concurrency Theory*, volume 140 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 36, 16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

19 Chase Ford, Harsh Beohar, Barbara König, Stefan Milius, and Lutz Schröder. Graded monads and behavioural equivalence games. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages [Article 61], 13. ACM, New York, 2022. doi:10.1145/3531130.3533374.

20 Chase Ford, Stefan Milius, and Lutz Schröder. Behavioural preorders via graded monads. In *Proceedings of the 2021 36th annual ACM/IEEE symposium on logic in computer science, LICS 2021, Rome, Italy, June 29 – July 2, 2021*, page 13. Piscataway, NJ: IEEE Press, 2021. Id/No 12. doi:10.1109/LICS52264.2021.9470517.

21 Jonas Forster, Lutz Schröder, Paul Wild, Harsh Beohar, Sebastian Gurke, and Karla Messing. Graded semantics and graded logics for Eilenberg-Moore coalgebras. In *Coalgebraic methods in computer science*, volume 14617 of *Lecture Notes in Comput. Sci.*, pages 114–134. Springer, Cham, 2024. doi:10.1007/978-3-031-66438-0_6.

22 Tobias Fritz. A presentation of the category of stochastic matrices, March 2009. arXiv:0902.2554v2.

23 Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. doi:10.1016/j.aim.2020.107239.

24 Marco Gaboardi, Shin-ya Katsumata, Dominic Orchard, Flavien Breuvart, and Tarmo Uustalu. Combining effects and coeffects via grading. In *ICFP'16—Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, pages 476–489. ACM, New York, 2016. doi:10.1145/2951913.2951939.

25 Marco Gaboardi, Shin-ya Katsumata, Dominic Orchard, and Tetsuya Sato. Graded Hoare logic and its categorical semantics. In *Programming Languages and Systems: 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings*, pages 234–263, Berlin, Heidelberg, 2021. Springer-Verlag. doi:10.1007/978-3-030-72019-3_9.

26 Bruno Gavranović. *Fundamental Components of Deep Learning: A Category-Theoretic Approach*. PhD thesis, University of Strathclyde, Glasgow, UK, 2024. Department of Computer and Information Sciences, Thesis identifier: T16859. doi:10.48730/795r-wn11.

27 Amar Hadzihasanovic. A diagrammatic axiomatisation for qubit entanglement. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015)*, pages 573–584. IEEE Computer Soc., Los Alamitos, CA, 2015. doi:10.1109/LICS.2015.59.

28 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. In *LICS '18—33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, page [10 pp.]. ACM, New York, 2018. doi:10.1145/3209108.3209131.

29 Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. Completeness of the ZX-calculus. *Log. Methods Comput. Sci.*, 16(2):Paper No. 11, 72, 2020. doi:10.23638/LMCS-16(2:11)2020.

30 Shin-ya Katsumata. Parametric effect monads and semantics of effect systems. *SIGPLAN Not.*, 49(1):633–645, January 2014. doi:10.1145/2578855.2535846.

31 Shin-ya Katsumata, Dylan McDermott, Tarmo Uustalu, and Nicolas Wu. Flexible presentations of graded monads. *Proc. ACM Program. Lang.*, 6(ICFP), August 2022. doi:10.1145/3547654.

32 Satoshi Kura. Graded algebraic theories. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures*, pages 401–421, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-45231-5_21.

33 F. W. Lawvere. Functorial semantics of algebraic theories. *Proc. Natl. Acad. Sci. USA*, 50:869–872, 1963. doi:10.1073/pnas.50.5.869.

34 Jack Liell-Cock and Sam Staton. Compositional imprecise probability: A solution from graded monads and Markov categories. *Proc. ACM Program. Lang.*, 9(POPL), January 2025. doi:10.1145/3704890.

35 Gabriele Lobbia, Wojciech Różowski, Ralph Sarkis, and Fabio Zanasi. Quantitative monoidal algebra: Axiomatising distance with string diagrams, 2025. arXiv:2410.09229.

36 Rory B. B. Lucyshyn-Wright. V-graded categories and V-W-bigraded categories: Functor categories and bifunctors over non-symmetric bases, 2025. arXiv:2502.18557.

37 Dylan McDermott and Tarmo Uustalu. Flexibly graded monads and graded algebras. In Ekaterina Komendantskaya, editor, *Mathematics of Program Construction*, pages 102–128, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-16912-0_4.

38 Paul-André Melliès. Parametric monads and enriched adjunctions, 2012. Preprint. URL: https://www.irif.fr/~mellies/tensorial-logic/8-parametric-monads-and-enriched-adjunctions.pdf.

39 Stefan Milius, Dirk Pattinson, and Lutz Schröder. Generic trace semantics and graded monads. In *6th Conference on Algebra and Coalgebra in Computer Science*, volume 35 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 253–269. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPICS.CALCO.2015.253.

40 Alan Mycroft, Dominic Orchard, and Tomas Petricek. Effect systems revisited – control-flow algebra and semantics. In *Semantics, logics, and calculi. Essays dedicated to Hanne Riis Nielson and Flemming Nielson on the occasion of their 60th birthdays*, pages 1–32. Cham: Springer, 2016. doi:10.1007/978-3-319-27810-0_1.

41 David Jaz Myers. String diagrams for double categories and equipments, 2018. arXiv:1612.02762.

42 Robin Piedeleu, Mateo Torres-Ruiz, Alexandra Silva, and Fabio Zanasi. A complete axiomatisation of equivalence for discrete probabilistic programming, 2024. doi:10.48550/arXiv.2408.14701.

43 Robin Piedeleu and Fabio Zanasi. A Finite Axiomatisation of Finite-State Automata Using String Diagrams. *Logical Methods in Computer Science*, Volume 19, Issue 1, February 2023. doi:10.46298/lmcs-19(1:13)2023.

44 Robin Piedeleu and Fabio Zanasi. An introduction to string diagrams for computer scientists. *CoRR*, abs/2305.08768, 2023. doi:10.48550/arXiv.2305.08768.

45 Kate Ponto and Michael Shulman. Shadows and traces in bicategories. *J. Homotopy Relat. Struct.*, 8(2):151–200, 2013. doi:10.1007/s40062-012-0017-0.

46 Boldizsár Poór, Quanlong Wang, Razin A. Shaikh, Lia Yeh, Richie Yeung, and Bob Coecke. Completeness for arbitrary finite dimensions of ZXW-calculus, a unifying calculus. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 14. IEEE Comput. Soc. Press, Los Alamitos, CA, 2023.

47 Takahiro Sanada. Category-graded algebraic theories and effect handlers. *Electronic Notes in Theoretical Informatics and Computer Science*, Volume 1 - Proceedings of MFPS XXXVIII, February 2023. doi:10.46298/entics.10491.

48 P. Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, volume 813 of *Lecture Notes in Phys.*, pages 289–355. Springer, Heidelberg, 2011. doi:10.1007/978-3-642-12821-9_4.

49 Dario Stein, Fabio Zanasi, Robin Piedeleu, and Richard Samuelson. Graphical quadratic algebra, 2024. doi:10.48550/arXiv.2403.02284.

50 Alejandro Villoria, Henning Basold, and Alfons Laarman. Enriching diagrams with algebraic operations. In *Foundations of software science and computation structures. 27th international conference, FOSSACS 2024, held as part of the European joint conferences on theory and practice of software, ETAPS 2024, Luxembourg City, Luxembourg, April 6–11, 2024. Proceedings. Part I*, pages 121–143. Cham: Springer, 2024. doi:10.1007/978-3-031-57228-9_7.

51 Wikipedia. Boy or girl paradox — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Boy%20or%20girl%20paradox&oldid=1269474033, 2025. [Online; accessed 27-February-2025]. URL: https://en.wikipedia.org/wiki/Boy_or_girl_paradox.

52 Richard James Wood. *Indical Methods for Relative Categories*. PhD thesis, Dalhousie University, 1976. Available at https://dam-oclc.bac-lac.gc.ca/download?is_thesis=1&oclc_number=1340918297&id=52f7c7de-6478-456b-93ef-27a04de3696b&fileName=NK31546.PDF.

53 Fabio Zanasi. *Interacting Hopf Algebras: the theory of linear systems*. Theses, ENS de Lyon, October 2015. Available at https://theses.hal.science/tel-01218015. URL: https://theses.hal.science/tel-01218015.

A Appendix to Section 2

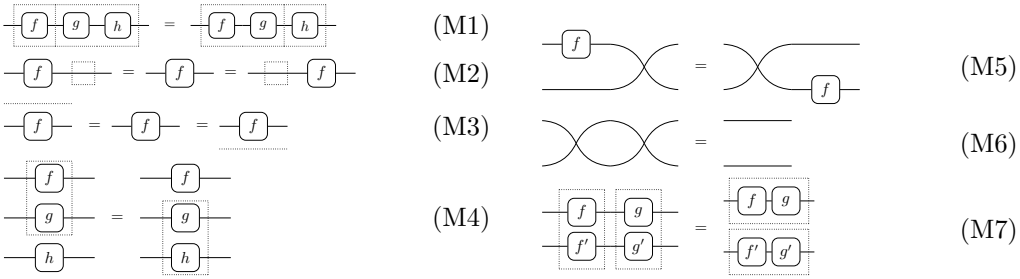


Figure A.1 Laws of symmetric strict monoidal categories.

B Appendix to Section 3

We provide the general details of the **Para** construction. Given a symmetric monoidal action $\bullet : \mathbb{G} \times \mathbf{C} \to \mathbf{C}$, $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ is defined as follows.

- Its objects are those of $\mathbf{C}$.
- The set of a-graded morphisms from X to Y is $\mathbf{Para}(\mathbb{G}, \mathbf{C})_a(X, Y) = \mathbf{C}(a \bullet X, Y)$.

- For every morphism between grades $t : b \to a$, and graded morphism $f : X \xrightarrow{a} Y$,

$$t \star f \coloneqq b \bullet X \xrightarrow{t \bullet \mathrm{id}_X} a \bullet X \xrightarrow{f} Y. \tag{15}$$

- The composition of two graded morphisms $f : X \xrightarrow{a} Y$ and $g : Y \xrightarrow{b} Z$ is given by

$$f \fatsemi g \coloneqq (a+b) \bullet X \cong b \bullet (a \bullet X) \xrightarrow{\mathrm{id}_b \bullet f} b \bullet Y \xrightarrow{g} Z. \tag{16}$$

- The identity morphism $\mathrm{id}_X : X \xrightarrow{0} X$ is $0 \bullet X \cong X \xrightarrow{\mathrm{id}_X} X$.
- The monoidal product of objects in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ is inherited from the monoidal product of objects in $\mathbf{C}$.
- The monoidal product of $f : X \xrightarrow{a} Y$ and $g : X' \xrightarrow{b} Y'$ is given by (the isomorphism is obtained from the structure of a symmetric monoidal actegory, it is called the *interchanger* in [14])

$$f \otimes g \coloneqq (a+b) \bullet (X \otimes X') \cong (a \bullet X) \otimes (b \bullet X') \xrightarrow{f \otimes g} Y \otimes Y'. \tag{17}$$

- The symmetries are inherited from the symmetries of $\mathbf{C}$.

By our hypotheses on the action $\bullet$, the grade 0 acts trivially (via $0 \bullet X \cong X$), so the underlying SMC of $\mathbf{Para}(\mathbb{G}, \mathbf{C})$ (i.e. the SMC of 0-graded morphisms) is isomorphic to $\mathbf{C}$.

Proof of Lemma 24. Formally, T_0 is defined by

$$\Sigma_0 = \{g : n \to m \mid g : n \xrightarrow{0} m \in \Sigma\} \quad E_0 = \{f = f' \mid f = f' \in E_{\mathsf{h}} \text{ and } f, f' : n \xrightarrow{0} m\}.$$

Let $\mathcal{S}_{\mathsf{T}_0}$ be the prop freely generated by T_0 and $U\mathcal{S}_{\mathsf{T}}$ denote the underlying prop of $\mathcal{S}_{\mathsf{T}}$. Also, let i be the map sending every generator in Σ_0 to its 0-graded counterpart in $U\mathcal{S}_{\mathsf{T}}$. We will show that i canonically extends to an isomorphism of SMCs $\mathcal{S}_{\mathsf{T}_0} \cong U\mathcal{S}_{\mathsf{T}}$.

Since $U\mathcal{S}_{\mathsf{T}}$ is an SMC, there is an extension of i to monoidal Σ_0-terms (that we abusively denote with i) $i : \mathcal{S}_{\Sigma_0} \to U\mathcal{S}_{\mathsf{T}}$. By the universal property of $\mathcal{S}_{\mathsf{T}_0}$, to check that i factors through $\mathcal{S}_{\mathsf{T}_0}$, it suffices to show that the equations in E_0 hold in $U\mathcal{S}_{\mathsf{T}}$ after applying i. This is immediate from the fact that the equations in E_0 are contained in E_{h}.

Next, in order to show the factored functor $i^* : \mathcal{S}_{\mathsf{T}_0} \to U\mathcal{S}_{\mathsf{T}}$ is an isomorphism, we need to show that any 0-graded term of $\mathcal{S}_{\mathsf{T}}$ is in the image of i^*, and that whenever $f = g$ is in the closure $\overline{E_{\mathsf{h}}}$, then $f = g$ is derivable from E_0. This boils down to comparing the restriction of Figure 1 to 0-graded terms with the axioms of SMCs to find they are exactly the same. In particular, sequential and parallel compositions and regrading of terms of grades other than 0 cannot yield 0-graded terms, thus all 0-graded terms are obtained by sequential and parallel composition of 0-graded generators, identities, and symmetries. This implies that i^* is full. Moreover, the equations in $\overline{E_{\mathsf{h}}}$ between terms of grade bigger than 0 cannot influence the equations between 0-graded terms, and this implies i^* is faithful. ◀

C Appendix to Section 4

We fill in details that were omitted in proofs in the main body. Recall that ; denotes composition in $\mathbb{G}$ or $\mathbf{C}$, while $\fatsemi$ denotes composition in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$. For simplicity, we will not draw thick wires, but recall that $\overset{a}{\sqcap}$ has a grading wires and ak output wires, where $k \coloneqq 1 \bullet 0$ determines the whole action of $\bullet$.

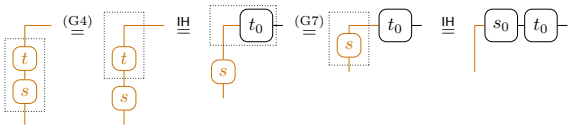
Let us first show that $a \bullet n = n + ak$ for any $a, n \in \mathbb{N}$. This relies on the following three properties that are true in any symmetric monoidal action (see [14]).

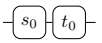
$$A \bullet (X \otimes Y) \overset{*}{\cong} X \otimes (A \bullet Y) \qquad (A \otimes B) \bullet (X \otimes Y) \overset{**}{\cong} (A \bullet X) \otimes (B \bullet Y) \qquad I \bullet X \overset{***}{\cong} X$$

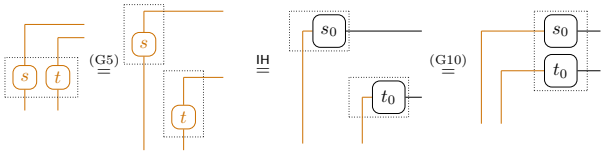
- For any $n \in \mathbb{N}$, we have $a \bullet n = a \bullet (n+0) \overset{*}{=} n + a \bullet 0 = n + (a \bullet 0)$.
- For any $a \in \mathbb{N}$, we have $a + 1 \bullet 0 = (a+1) \bullet (0+0) \overset{**}{=} a \bullet 0 + 1 \bullet 0$. Thus, by initializing an induction with the base case $1 \bullet 0 = k$, we find that $a \bullet 0 = ak$ for all $a > 0$.
- Since $0 \bullet n \overset{***}{=} n$, we conclude that for all $a, n \in \mathbb{N}$, $a \bullet n = n + ak$.

Proof of Lemma 25. Since $\mathbb{G}$ is presented by a monoidal theory $(\Sigma_{\mathsf{v}}, E_{\mathsf{v}})$, we can proceed by induction on a representative monoidal Σ_{v}-term of . The base cases (when t is a generator) are handled by the axioms that are in E_{h} (10).

For sequential composition, we use the induction hypothesis (indicated by IH) twice in the following derivation.



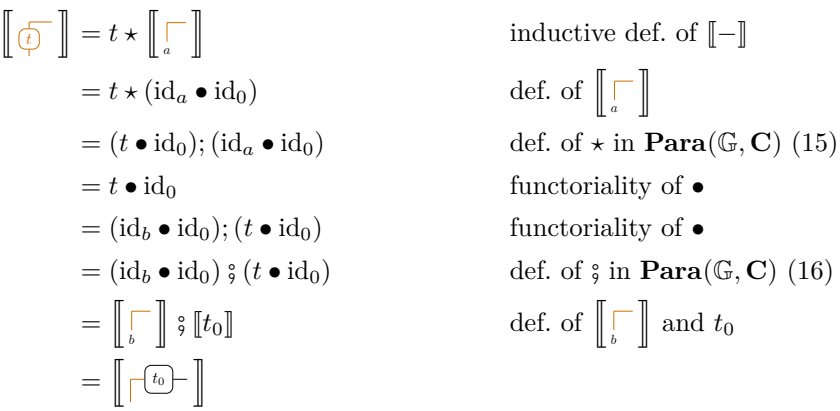
We can show that  is in the equivalence class of $(s;t) \bullet \mathrm{id}_0$. Indeed, by functoriality of $\bullet$, we have $(s;t) \bullet \mathrm{id}_0 = (s;t) \bullet (\mathrm{id}_0; \mathrm{id}_0) = (s \bullet \mathrm{id}_0); (t \bullet \mathrm{id}_0)$. Thus, if s_0 and t_0 are representatives of $s \bullet \mathrm{id}_0$ and $t \bullet \mathrm{id}_0$ respectively, then $s_0; t_0$ is a representative of $(s;t) \bullet \mathrm{id}_0$. Moreover, for any other term u_0 in that equivalence class, a proof of $s_0; t_0 = u_0$ in the monoidal theory presenting $\mathbf{C}$ can be adapted to a proof that $s_0; t_0 = u_0$ belongs to $\overline{E_{\mathsf{h}}}$. For parallel composition, we also apply the induction hypothesis twice.



Then, the rest of the argument follows as above (but instead of functoriality of $\bullet$, we use the fact that it is monoidal). ◀

Proof of Theorem 27. We expand on a small derivation in the proof.

To show the axioms coming from (10) hold after applying $[\![-]\!]$, we show that for any for any morphism $t : b \to a \in \mathbb{G}$, we have the following derivation.



We expand on the proof that $[\![-]\!]$ is fully faithful.

For faithfulness, suppose that $f, g : n \xrightarrow{a} m$ are Σ-terms such that $[\![f]\!] = [\![g]\!]$. We will show that $f = g \in \overline{E_{\mathsf{h}}}$. Observe that the proofs of Lemmas 25 and 26 did not rely on the equations in E_{h}, only on equations in Figure 1 that are valid in any graded SMCs. Therefore, we automatically have

Let us unroll some definitions to study precomposition by $[\![\overline{\ulcorner}]\!]$:

$$\begin{aligned}
[\![\overline{\ulcorner}]\!] \fatsemi - &= (\mathrm{id}_n \otimes (\mathrm{id}_a \bullet \mathrm{id}_0)) \fatsemi - && \text{def. of } [\![-]\!] \\
&= (\mathrm{id}_0 \bullet (\mathrm{id}_n \otimes (\mathrm{id}_a \bullet \mathrm{id}_0))); - && \text{def. of } \fatsemi \text{ in } \mathbf{Para}(\mathbb{G}, \mathbf{C}) \ (16) \\
&= \mathrm{id}_{n+ak}; - && \text{functoriality of } \bullet \text{ and } \otimes
\end{aligned}$$

This shows that pre-composition by $[\![\overline{\ulcorner}]\!]$ is injective, and hence, with the previous equation, we obtain $[\![f_0]\!] = [\![g_0]\!]$. Since $[\![-]\!]$ is faithful on 0-graded terms, we conclude that $f_0 = g_0 \in \overline{E_{\mathsf{h}}}$, thus $f = g \in \overline{E_{\mathsf{h}}}$.

For fullness, let $u : n \xrightarrow{a} m$ be a morphism in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$. We will show there is a term f_u satisfying $[\![f_u]\!] = u$. Viewing it inside $\mathbf{C}$, u is a morphism $n + ak \to m$, which we can see as a 0-graded morphism $u_0 : n + ak \xrightarrow{0} m$ in $\mathbf{Para}(\mathbb{G}, \mathbf{C})$. Since $[\![-]\!]$ is full on 0-graded morphisms, there is a 0-graded Σ-term f_{u_0} such that $[\![f_{u_0}]\!] = u_0$. Now, we define f_u by precomposing f_{u_0} with $\overline{\ulcorner}$, and we find (as desired)

$$[\![f_u]\!] = [\![\overline{\ulcorner}]\!] \fatsemi [\![f_{u_0}]\!] = \mathrm{id}_{n+ak}; u = u. \qquad \blacktriangleleft$$

D Appendix to Section 6

$$\frac{}{\Gamma, x : \tau \vdash x : \tau \,\&\, 0}\ \textsc{Var} \qquad \frac{\Gamma \vdash e : \tau_1 \,\&\, a}{\Gamma, x : \tau_0 \vdash e : \tau_1 \,\&\, a}\ \textsc{Weak} \qquad \frac{\Gamma \vdash e : \tau_1 \otimes \tau_2 \,\&\, a}{\Gamma \vdash \mathtt{fst}\ e : \tau_1 \,\&\, a}\ \textsc{Fst}$$

$$\frac{p \in [0,1]}{\Gamma \vdash \mathtt{flip}(p) : \mathbb{B} \,\&\, 0}\ \textsc{Flip} \qquad \frac{}{\Gamma \vdash \mathtt{knight} : \mathbb{B} \,\&\, 1}\ \textsc{Knight} \qquad \frac{\Gamma \vdash e : \tau_1 \otimes \tau_2 \,\&\, a}{\Gamma \vdash \mathtt{snd}\ e : \tau_2 \,\&\, a}\ \textsc{Snd}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \,\&\, a \qquad \Gamma \vdash e_2 : \tau_2 \,\&\, b}{\Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \otimes \tau_2 \,\&\, a + b}\ \textsc{Pair} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \,\&\, a \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau \,\&\, b}{\Gamma \vdash \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2 : \tau \,\&\, a + b}\ \textsc{Let}$$

$$\frac{\Gamma \vdash e : \mathbb{B} \,\&\, a \qquad \Gamma \vdash e_1 : \tau \,\&\, b \qquad \Gamma \vdash e_2 : \tau \,\&\, c}{\Gamma \vdash \mathtt{if}\ e\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 : \tau \,\&\, a + b + c}\ \textsc{If}$$

Figure D.1 Typing rules for a discrete imprecise probabilistic programming language.

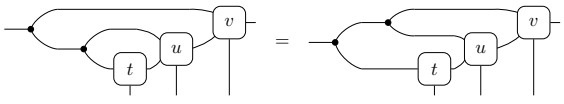
Proof of Theorem 33. First of all, we specify the contexts in which the equations hold.

$$\Gamma \vdash t : \tau_1 \,\&\, a \qquad \Gamma, x : \tau \vdash u : \tau_2 \,\&\, b \qquad \Gamma, y : \tau_2 \vdash v : \tau \,\&\, c$$
$$\mathtt{let}\ x = t\ \mathtt{in}\ (\mathtt{let}\ y = u\ \mathtt{in}\ v) \approx \mathtt{let}\ y = (\mathtt{let}\ x = t\ \mathtt{in}\ u)\ \mathtt{in}\ v \qquad \text{(associativity)}$$

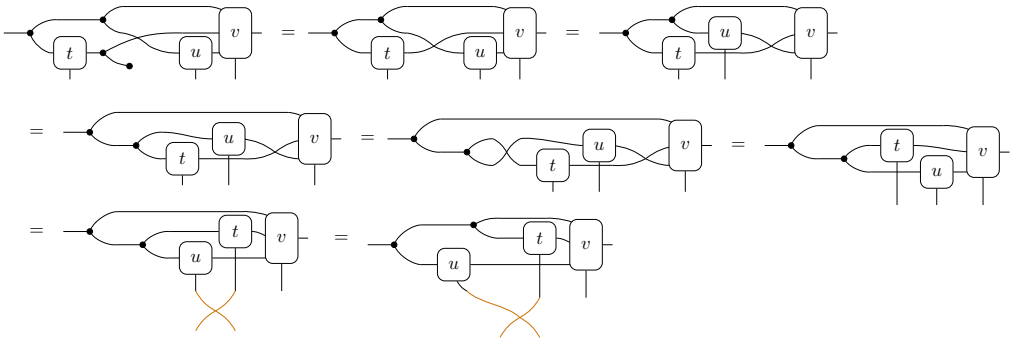
$$\Gamma \vdash t : \tau_1 \,\&\, a \qquad \Gamma \vdash u : \tau_2 \,\&\, b \qquad \Gamma, x : \tau_1, y : \tau_2 \vdash v : \tau \,\&\, c$$
$$\mathtt{let}\ x = t\ \mathtt{in}\ (\mathtt{let}\ y = u\ \mathtt{in}\ v) \approx \mathtt{let}\ y = u\ \mathtt{in}\ (\mathtt{let}\ x = t\ \mathtt{in}\ v) \qquad \text{(commutativity)}$$
$$\mathtt{let}\ x = t\ \mathtt{in}\ u \approx u \qquad \text{(weakening)}$$

The proof is purely an exercise in diagrammatic reasoning. We omit the recurring semantics brackets $[\![-]\!]$ in the diagrams to improve readability.

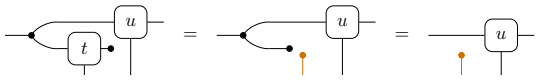
Associativity holds by associativity of the copy generator (and its n-ary version).



Commutativity holds (up to regrading) by the following derivation relying on commutativity of the copy generator (and its n-ary version).



Weakening holds (up to regrading) by the following derivation relying on the properties of the discard generator (and its n-ary version): it is unital for the copy generator, it annihilates any 0-graded term (i.e. $f \bullet = \bullet$), and it slides through the generator ⌐ by (14).



◀