

EPTCS 429

Proceedings of the
**Seventh International Conference on
Applied Category Theory 2024**

Oxford, United Kingdom, 17 - 21 June 2024

Edited by: Michael Johnson and David Jaz Myers

Published: 25th September 2025
DOI: 10.4204/EPTCS.429
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Table of Contents	i
Preface	iii
Pattern Runs on Matter: The Free Monad Monad as a Module over the Cofree Comonad Comonad. <i>Sophie Libkind and David I. Spivak</i>	1
Abstract Kleisli Structures on 2-categories	29
<i>Adrian Miranda</i>	
Learners are Almost Free Compact Closed	49
<i>Mitchell Riley</i>	
Combs, Causality and Contractions in Atomic Markov Categories	63
<i>Dario Stein and Márk Széles</i>	
Presenting Profunctors	88
<i>Gabriel Goren-Roig, Joshua Meyers and Emilio Minichiello</i>	
Reverse Faà di Bruno’s Formula for Cartesian Reverse Differential Categories	115
<i>Aaron Biggin and Jean-Simon Pacaud Lemay</i>	
Higher-Order DisCoCat (Peirce-Lambek-Montague semantics)	130
<i>Alexis Toumi and Giovanni de Felice</i>	
Copy-composition for Probabilistic Graphical Models	146
<i>Toby St Clere Smithe</i>	
Representing Knowledge and Querying Data using Double-Functorial Semantics	174
<i>Michael Lambert and Evan Patterson</i>	
A Category Theoretic Approach to Approximate Game Theory	190
<i>Neil Ghani</i>	
Fibrational Perspectives on Determinization of Finite-State Automata	203
<i>Thea Li</i>	
Generalized Gradient Descent is a Hypergraph Functor	217
<i>Tyler Hanks, James Fairbanks and Matthew Klawonn</i>	
ViCAR: Visualizing Categories with Automated Rewriting in Coq	234

<i>Bhakti Shah, Willam Spencer, Laura Zielinski, Ben Caldwell, Adrian Lehmann and Robert Rand</i>	
A Coalgebraic Model of Quantum Bisimulation	249
<i>Lorenzo Ceragioli, Elena Di Lavore, Giuseppe Lomurno and Gabriele Tedeschi</i>	
Reinforcement Learning in Categorical Cybernetics	270
<i>Jules Hedges and Riu Rodríguez Sakamoto</i>	
Organizing Physics with Open Energy-Driven Systems	287
<i>Matteo Capucci, Owen Lynch and David I. Spivak</i>	

Preface

David Jaz Myers

Topos Institute
Oxford, UK

davidjaz@topos.institute

Michael Johnson

Macquarie University
Sydney, Australia

mqmikej@gmail.com

The Seventh International Conference on Applied Category Theory took place at the University of Oxford on 17 - 21 June 2024, following the previous meetings at Leiden (2018), Oxford (2019), MIT (2020, fully online), Cambridge (2021), Strathclyde (2022), and Maryland (2023). It was preceded by the Adjoint School 2024 (10 - 14 June), a collaborative research event in which junior researchers worked under the mentorship of experts. The conference comprised 49 contributed talks, a poster session, an industry showcase session, and a session where junior researchers who had attended the Adjoint School presented the results of their research at the school. Information regarding the conference may be found at <https://oxford2024.github.io/>.

ACT 2024 was a hybrid event, with physical attendees present in Oxford and other participants taking part online. All talks were streamed to Zoom with synchronous discussion on Zulip.

Submission to ACT 2024 had three tracks: extended abstracts, software demonstrations, and proceedings. The extended abstract and software demonstration submissions had a page limit of 2 pages, and the proceedings track had a page limit of 12 pages. Only papers submitted to the proceedings track were considered for publication in this volume. In total, there were 72 submissions, of which 49 were accepted for presentation and 16 for publication in this volume. Each submission received a review from three different members of the programming committee, and papers were selected based on discussion and consensus by these reviewers.

The contributions to ACT 2024 ranged from pure to applied and included contributions in a wide range of disciplines in science and engineering. ACT 2024 included talks in classical mechanics, quantum physics, probability theory, linguistics, decision theory, machine learning, epidemiology, thermodynamics, engineering, and logic. Many of the submissions had software demonstrating their work or represented work done in collaboration with industry or a scientific organization. The industry session included five invited talks by practitioners using category theory in private enterprise.

We are grateful to Sam Staton, Lara Hartley, and their team at Oxford who contributed valuably to the organization and smooth running of the event. We are also grateful to the organizers of the co-located MFPS conference, Valeria de Paiva and Alex Simpson, who cooperated so graciously in coordinating joint invited speakers and facilitating the joint attendance of registrants.

ACT 2024 was sponsored by Conexus, Cybercat Institute, Symbolica, Topos Institute, Viete, and the University of Oxford.

David Jaz Myers and Michael Johnson
Chairs of the ACT 2024 Programme Committee

Pattern Runs on Matter: The Free Monad Monad as a Module over the Cofree Comonad Comonad

Sophie Libkind

Topos Institute
Berkeley, CA, USA

sophie@topos.institute

David I. Spivak

Topos Institute
Berkeley, CA, USA

david@topos.institute

Interviews run on people, programs run on operating systems, voting schemes run on voters, games run on players. Each of these is an example of the abstraction *pattern runs on matter*. Pattern determines the decision tree that governs how a situation can unfold, while matter responds with decisions at each juncture.

In this article, we will give a straightforward and concrete construction of the free monad monad for $(\mathbf{Poly}, \triangleleft, y)$, the category of polynomial functors with the substitution monoidal product. Although the free monad has been well-studied in other contexts, the construction we give is streamlined and explicitly illustrates how the free monad represents terminating decision trees. We will also explore the naturally arising interaction between the free monad and cofree comonad. Again, while the interaction itself is known, the perspective we take is the free monad as a module over the cofree comonad. Lastly, we will give four applications of the module action to interviews, computer programs, voting, and games. In each example, we will see how the free monad represents pattern, the cofree comonad represents matter, and the module action represents *runs on*.

1 Introduction

The etymology of matter and pattern are “mother” and “father”; this pair of terms offers a very basic sense in which to carve up the world. Like two parents, matter and pattern represent a fundamental dichotomy: matter is the pure material, unconcerned with our ideas about it; pattern is pure structure, unconcerned with what substantiates it. And yet one may have an intuitive sense that pattern “runs on”—must be instantiated in—matter. In this paper, we show that this idea matches both our intuition and the mathematics of a module structure by which free monads are a module over cofree comonads.

Intuitively, interviews, programs, voting schemes, and games represent patterns. But what is an interview without a person to be interviewed; what is a program without an operating system to run it on; what is a voting scheme without voters to do the voting; what is a game without players to play it? In each case, the pattern runs on a material substrate.

In this paper we give an account for this intuition in terms of free monads m and cofree comonads c on polynomial functors. We show that for any $p, q : \mathbf{Poly}$, there is a natural map

$$\Xi_{p,q} : m_p \otimes c_q \rightarrow m_{p \otimes q}. \quad (1)$$

In fact, this map gives m the structure of a c -module, in a precise sense; see Section 3.2. The free monad m_p represents patterns as terminating decision tree with shape p , the cofree comonad c_q represents matter as infinite behaviors trees of shape q , and the interaction law $\Xi_{p,q}$ interprets “runs on”.

Consider the behavior of a Moore machine, which transforms lists of A -inputs into lists of B -outputs for sets A, B . This behavior is determined by an element $b : y \rightarrow c_{By^A}$ of the terminal coalgebra on the polynomial $q := By^A$. But how does one apply this behavior to a list of A ’s? The latter is a map

$\ell: y \rightarrow m_{Ay}$. To actually apply the behavior b to the list ℓ , we use an obvious map $\varphi: Ay \otimes By^A \rightarrow By$ and the module structure (1) to obtain a list of B 's.

$$y \cong y \otimes y \xrightarrow{\ell \otimes b} m_{Ay} \otimes c_{By^A} \xrightarrow{\Xi_{Ay, By^A}} m_{Ay \otimes By^A} \xrightarrow{m_\varphi} m_{By} \quad (2)$$

Whereas patterns start and end, matter is never destroyed. One can think of m_p as the type of terminating programs and of c_q as the type of operating systems or online algorithms. This intuition is captured by the fact that elements of the free monad m_p are “well-founded trees” [1, 3]—in the case p is finitary, a wellfounded tree is one with finite height—whereas elements of the cofree comonad c_q are generally non-wellfounded, e.g. infinite in height even for finitary q . The module map $\Xi_{p,q}$ pairs the wellfounded tree with the non-wellfounded tree, following the shape of the wellfounded one; for example the list of B 's in (2) will have exactly the same length as the list of A 's.

Related work. A construction of the free monad for much more general (than polynomial) endofunctors was given by Kelly in [5]. The treatment was greatly simplified by Shulman and others in [10]. The case of polynomial endofunctors is more restrictive, and as such has a far more straightforward construction.

A monad-comonad interaction law, again in more general settings, was described in [4]. This paper structures interaction laws in an interesting and useful but somehow ad hoc way, as a category whose objects are triples (T, C, f) , where T is a monad, C is a comonad, and $f: T \otimes C \rightarrow \text{id}$ is a map satisfying certain natural conditions, and whose morphisms are maps $T \rightarrow T'$ and $C' \rightarrow C$ satisfying certain constraints.

Contributions.

1. Simple concrete constructions of both free monads and cofree comonads in $(\mathbf{Poly}, \triangleleft, y)$.
2. A proof that the free monad is a module over the cofree comonad.
3. Four applications of this module, each having the form *pattern runs on matter*.

For these, see Definition 2.3 and Proposition 3.1; Theorem 3.4; and Section 4, respectively. We also give a simplified definition of the “dual” of a polynomial functor that specializes the notion given in [4], and we generalize it to be functorial in a monad t ; see (6).

Notation. We often denote the identity on an object x by the object name itself rather than id_x . We denote the cardinality of a set X by $\#X$. If $L \dashv R$ is an adjunction, we denote it

$$C \begin{array}{c} \xrightarrow{L} \\ \Rightarrow \\ \xleftarrow{R} \end{array} \mathcal{D}$$

so that the 2-cell shown indicates the direction of both the unit $C \rightarrow R \circ L$ and the counit $L \circ R \rightarrow \mathcal{D}$.

Acknowledgements. The authors thanks Harrison Grodin and Brandon Shapiro for many useful conversations. This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA9550-20-1-0348 and FA9550-23-1-0376.

1.1 Preliminaries

Although we will assume basic familiarity with the category **Poly** of univariate polynomial functors on **Set** and natural transformations between them, we will begin by clarifying notation. We write a polynomial functor $p : \mathbf{Poly}$ as $p = \sum_{P:p(1)} y^{p[P]}$ where $p(1)$ are the **positions** of p and $p[P]$ are the **directions** at the position P . With this notation, a map $\phi : p \rightarrow q$ in **Poly** consists of

- A function on positions $\phi(1) : p(1) \rightarrow q(1)$.
- For each position $P : p(1)$, a function backwards on directions $\phi_P^\# : q[\phi_1(P)] \rightarrow p[P]$.

A map $\phi : p \rightarrow q$ in **Poly** is **cartesian** if for every position $P : p(1)$, the map on directions $\phi_P^\#$ is an isomorphism (equivalently, when all ϕ 's naturality squares are pullbacks).

There are many monoidal products in **Poly** [7], however in this article we focus on the Dirichlet and substitution products. The substitution product has unit y and is defined by

$$p \triangleleft q = \sum_{P:p(1)} \prod_{p[P]} \sum_{Q:q(1)} \prod_{q[Q]} y$$

Using the distributive law, a position in $p \triangleleft q$ is a position P in p and for every direction in $p[P]$ a position in q . It will be useful to consider the corolla forest view on polynomials and the $\triangleleft$ -monoidal product, since these will be the building blocks of the decision trees represented by the free monad monad.

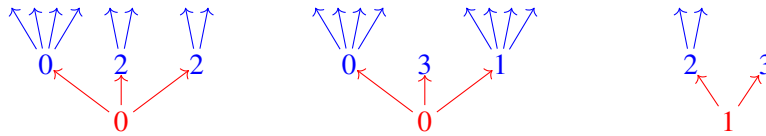
Consider $p = y^3 + y^2$. We can notate p using a corolla forest in which each position $P : p(1)$ is a tree with $p[P]$ -many branches. The root of each tree is labeled by its corresponding position and branches correspond to directions.



Likewise, the polynomial $q = 2y^4 + y^2 + 1$ is notated by the corolla forest:



Then the positions of $p \triangleleft q$ can be represented by trees of height 2, whose building blocks are the trees corresponding to the positions of p and q . In particular, a position of $p \triangleleft q$ is a position of p and for each branch, a position of q . The directions are the number of dangling leaves. Here are three examples of positions in $p \triangleleft q$ with eight, eight, and two directions respectively.



2 The Free Monad Monad

For a polynomial p , we will define the free monad m_p through transfinite induction [10] by defining polynomials $p_{(\alpha)}$ for each ordinal α and inclusions $p_{(\alpha)} \rightarrow p_{(\beta)}$ for ordinals $\alpha < \beta$.

- Base case: Define $p_{(0)} := y$.

- For successor ordinals $\alpha + 1$: Define $p_{(\alpha+1)} := y + p \triangleleft p_{(\alpha)}$. For the inclusions define $\iota_{(0)} : y \rightarrow y + p$ to be the inclusion; note that it is cartesian. Then, given $\iota_{(\alpha)}$, define $\iota_{(\alpha+1)}$ to be

$$p_{(\alpha+1)} = y + p \triangleleft p_{(\alpha)} \xrightarrow{y+p \triangleleft \iota_{(\alpha)}} y + p \triangleleft p_{(\alpha+1)} = p_{(\alpha+2)}.$$

which is also cartesian.

- For limit ordinals α : Define $p_{(\alpha)} := \text{colim}_{\alpha' < \alpha} p_{(\alpha')}$. For each $\alpha' < \alpha$, let $\iota_{(\alpha')} : p_{(\alpha')} \rightarrow p_{(\alpha)}$ be the natural inclusion. These are cartesian by Proposition A.5. Lastly, we define the inclusion $\iota_{(\alpha)} : p_{(\alpha)} \rightarrow p_{(\alpha+1)}$ to be induced by the cocone of maps which are defined for $\alpha' < \alpha$ by

$$p_{(\alpha')} \xrightarrow{\iota_{(\alpha')}} p_{(\alpha'+1)} = y + p \triangleleft p_{(\alpha')} \xrightarrow{y+p \triangleleft \iota_{(\alpha')}} y + p \triangleleft p_{(\alpha)} = p_{(\alpha+1)}. \quad (3)$$

For ease of notation, we will also use $\iota_{(\alpha)} : p_{(\alpha)} \rightarrow p_{(\beta)}$ to represent the composite of inclusions for any pair of ordinals $\alpha < \beta$. We will also define $\lambda_{(\alpha)} : p \triangleleft p_{(\alpha)} \rightarrow p_{(\alpha+1)}$ to be the coproduct inclusion.

Definition 2.1. For an ordinal κ , a polynomial p is called κ -**small** if for all κ -filtered categories $\mathcal{J}$ and all diagrams $Q : \mathcal{J} \rightarrow \mathbf{Set}$, the natural map

$$\text{colim}_{j: \mathcal{J}} (p \triangleleft Q_j) \rightarrow p \triangleleft \left(\text{colim}_{j: \mathcal{J}} Q_j \right)$$

is an isomorphism. $\diamond$

Remark 2.2. A polynomial $p : \mathbf{Poly}$ is κ -small if and only if all of its direction-sets have cardinality less than κ . It is called **finitary** if and only if it is ω -small. For every polynomial p , there is some κ for which p is κ -small. $\diamond$

Definition 2.3. If p is κ -small, then define $\mathfrak{m}_p := p_{(\kappa)}$. $\diamond$

We will show in Theorem 2.11 that $\mathfrak{m}_p$ is the free monad on p . Therefore, $\mathfrak{m}_p$ is unique and hence well-defined.

Think of the positions of $\mathfrak{m}_p$ as decision trees whose building blocks are the positions of p . For instance, suppose that a position $P : p(1)$ is a question and the directions $p[P]$ are possible answers to question P . Then y is a unique question with a unique answer, which we think of as “no further questions”. Then, consider the positions of $\mathfrak{m}_p$ that factor through $p_{(\alpha)}$. For example, the positions of $p_{(2)} = y + p \triangleleft (y + p)$ is either (1) no further questions or (2) a question and for each possible answer either no further questions or another question. Therefore $p_{(2)}$ represents interviews with *at most two questions*, and in general for finite i , $p_{(i)}$ represents interviews with at most i questions. The directions of $\mathfrak{m}_p$ represent the possible paths through the decision tree.

Example 2.4. For finitary p , the positions of $\mathfrak{m}_p$ are p -trees with finite height and whose directions are the dangling leaves. $\diamond$

Example 2.5. A finite y -tree is determined by its height and has one dangling leaf; thus, $\mathfrak{m}_y \cong \mathbb{N}y$. $\diamond$

2.1 Monad Structure on $\mathfrak{m}_p$

Let $\mathbf{Mod}(\mathbf{Poly})$ denote the category of monoids in $(\mathbf{Poly}, y, \triangleleft)$. Note that when viewed as endofunctors on $\mathbf{Set}$, a $\triangleleft$ -monoid is in fact a monad on $\mathbf{Set}$. Therefore we refer to the objects of $\mathbf{Mod}(\mathbf{Poly})$ as **$\triangleleft$ -monoids**, **polynomial monads**, or simply **monads**.

Next we will give a $\triangleleft$ -monoid structure on $\mathfrak{m}_p$. The unit $\eta : y \rightarrow \mathfrak{m}_p$ is defined to be the inclusion $\iota_{(0)} : y = p_{(0)} \rightarrow \mathfrak{m}_p$. The multiplication is more complicated so we begin with the following lemma.

Lemma 2.6. *For ordinals α, β there exist maps $\mu_{(\alpha, \beta)}: p_{(\alpha)} \triangleleft p_{(\beta)} \rightarrow p_{(\alpha+\beta)}$ such that for all $\alpha' < \alpha$ and $\beta' < \beta$ the following diagram commutes*

$$\begin{array}{ccc} p_{(\alpha')} \triangleleft p_{(\beta')} & \xrightarrow{\mu_{(\alpha', \beta')}} & p_{(\alpha'+\beta')} \\ \downarrow \iota_{(\alpha')} \triangleleft \iota_{(\beta')} & & \downarrow \iota_{(\alpha'+\beta')} \\ p_{(\alpha)} \triangleleft p_{(\beta)} & \xrightarrow{\mu_{(\alpha, \beta)}} & p_{(\alpha+\beta)} \end{array} \quad (4)$$

Proof. We define the maps $\mu_{(\alpha, \beta)}$ transinductively on α .

For $\alpha = 0$, $\mu_{(\alpha, \beta)}$ is the identity on $p_{(\beta)}$.

For successor ordinals $\alpha + 1$, suppose we have already defined $\mu_{(\alpha, \beta)}$. Note that

$$p_{(\alpha+1)} \triangleleft p_{(\beta)} = (y + p \triangleleft p_{(\alpha)}) \triangleleft p_{(\beta)} = p_{(\beta)} + p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta)}.$$

So we define $\mu_{(\alpha+1, \beta)}$ to be the copairing of the inclusion $\iota_{(\beta)}: p_{(\beta)} \rightarrow p_{(\alpha+\beta)}$ with the map

$$p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta)} \xrightarrow{p \triangleleft \mu_{(\alpha, \beta)}} p \triangleleft p_{(\alpha+\beta)} \xrightarrow{\lambda_{(\alpha+\beta)}} p_{(\alpha+\beta+1)}.$$

That the diagram in (4) commutes for $\alpha' = \alpha + 1$ can be shown inductively on α .

Next suppose that α is a limit ordinal and suppose we have defined $\mu_{(\alpha', \beta)}: p_{(\alpha')} \triangleleft p_{(\beta)} \rightarrow p_{(\alpha'+\beta)}$ for $\alpha' < \alpha$. Then we define $\mu_{(\alpha, \beta)}$ to be the composite

$$p_{(\alpha)} \triangleleft p_{(\beta)} = \left(\operatorname{colim}_{\alpha' < \alpha} p_{(\alpha')} \right) \triangleleft p_{(\beta)} \cong \operatorname{colim}_{\alpha' < \alpha} (p_{(\alpha')} \triangleleft p_{(\beta)}) \rightarrow \operatorname{colim}_{\alpha' < \alpha} p_{(\alpha'+\beta)} \rightarrow p_{(\alpha+\beta)}.$$

The second isomorphism follows from Proposition A.6. That these maps make the diagram in (4) commute is immediate. $\square$

This lemma implies that for ordinals $\alpha, \beta < \kappa$ the maps

$$p_{(\alpha)} \triangleleft p_{(\beta)} \xrightarrow{\mu_{(\alpha, \beta)}} p_{(\alpha+\beta)} \rightarrow \mathfrak{m}_p$$

form a cocone into $\mathfrak{m}_p$. Define the multiplication $\mu: \mathfrak{m}_p \triangleleft \mathfrak{m}_p \rightarrow \mathfrak{m}_p$ using the universal property:

$$\mathfrak{m}_p \triangleleft \mathfrak{m}_p = \left(\operatorname{colim}_{\alpha < \kappa} p_{(\alpha)} \right) \triangleleft \left(\operatorname{colim}_{\beta < \kappa} p_{(\beta)} \right) \cong \operatorname{colim}_{\alpha < \kappa} \operatorname{colim}_{\beta < \kappa} (p_{(\alpha)} \triangleleft p_{(\beta)}) \rightarrow \mathfrak{m}_p.$$

The second isomorphism follows from Proposition A.1 and Proposition A.6. Explicitly, given a decision tree and for each dangling leaf another decision tree, the multiplication $\mathfrak{m}_p \triangleleft \mathfrak{m}_p \rightarrow \mathfrak{m}_p$ glues these together into a single decision tree. From this definition and Lemma (2.6), the following lemma is immediate.

Lemma 2.7. *Let p be κ -small. Then for all $\alpha, \beta < \kappa$, the following diagram commutes.*

$$\begin{array}{ccc} p_{(\alpha)} \triangleleft p_{(\beta)} & \xrightarrow{\mu_{(\alpha, \beta)}} & p_{(\alpha+\beta)} \\ \downarrow \iota_{(\alpha)} \triangleleft \iota_{(\beta)} & & \downarrow \iota_{(\alpha+\beta)} \\ \mathfrak{m}_p \triangleleft \mathfrak{m}_p & \xrightarrow{\mu} & \mathfrak{m}_p \end{array}$$

As a predecessor to proving the associativity law for $(\mathfrak{m}_p, \eta, \mu)$ as a $\triangleleft$ -monoid, we present the following variant of Lemma 2.6 whose proof appears in Appendix B.

Lemma 2.8. *For ordinals α, β, γ , the following diagram commutes:*

$$\begin{array}{ccc} p(\alpha) \triangleleft p(\beta) \triangleleft p(\gamma) & \xrightarrow{p(\alpha) \triangleleft \mu(\beta, \gamma)} & p(\alpha) \triangleleft p(\beta + \gamma) \\ \mu(\alpha, \beta) \triangleleft p(\gamma) \downarrow & & \downarrow \mu(\alpha, \beta + \gamma) \\ p(\alpha + \beta) \triangleleft p(\gamma) & \xrightarrow{\mu(\alpha + \beta, \gamma)} & p(\alpha + \beta + \gamma) \end{array}$$

Proposition 2.9. *For every polynomial $p : \mathbf{Poly}$, there is a $\triangleleft$ -monoid structure on $\mathfrak{m}_p$, for which the unit and multiplication $\eta : y \rightarrow \mathfrak{m}_p$ and $\mu : \mathfrak{m}_p \triangleleft \mathfrak{m}_p \rightarrow \mathfrak{m}_p$ are defined as above.*

Proof. First we show the left unit law. Since $y \triangleleft \mathfrak{m}_p$ is isomorphic to $\text{colim}_{\beta < \kappa} (y \triangleleft p(\beta))$, the left unit law follows from Lemma 2.7 with $\alpha = 0$. Likewise, for the right unit law.

Second we show associativity. Due to the isomorphism

$$\mathfrak{m}_p \triangleleft \mathfrak{m}_p \triangleleft \mathfrak{m}_p \cong \text{colim}_{\alpha, \beta, \gamma < \kappa} (p(\alpha) \triangleleft p(\beta) \triangleleft p(\gamma)),$$

it suffices to show that for $\alpha, \beta, \gamma < \kappa$, the outer diagram in the following commutes.

$$\begin{array}{ccccc} p(\alpha) \triangleleft p(\beta) \triangleleft p(\gamma) & \xrightarrow{p(\alpha) \triangleleft \mu(\beta, \gamma)} & p(\alpha) \triangleleft p(\beta + \gamma) & \xrightarrow{\iota(\alpha) \triangleleft \iota(\beta + \gamma)} & \mathfrak{m}_p \triangleleft \mathfrak{m}_p \\ \mu(\alpha, \beta) \triangleleft p(\gamma) \downarrow & & \downarrow \mu(\alpha, \beta + \gamma) & & \downarrow \mu \\ p(\alpha + \beta) \triangleleft p(\gamma) & \xrightarrow{\mu(\alpha + \beta, \gamma)} & p(\alpha + \beta + \gamma) & & \\ \iota(\alpha + \beta) \triangleleft \iota(\gamma) \downarrow & & \searrow \iota(\alpha + \beta + \gamma) & & \downarrow \mu \\ \mathfrak{m}_p \triangleleft \mathfrak{m}_p & \xrightarrow{\mu} & & & \mathfrak{m}_p \end{array}$$

The upper left-hand square commutes by Lemma 2.8 and the other squares commute by Lemma 2.7. $\square$

2.2 The Monad $\mathfrak{m}_p$ is Free

Now that we have verified that $\mathfrak{m}_p$ is indeed a monad, we want to justify calling it the **free monad** by giving a left adjoint $\mathfrak{m}_- : \mathbf{Poly} \rightarrow \mathbf{Mod}(\mathbf{Poly})$ to the forgetful functor $U : \mathbf{Mod}(\mathbf{Poly}) \rightarrow \mathbf{Poly}$ which takes a $\triangleleft$ -monoid (q, η_q, μ_q) to its carrier q .

We begin by defining the action of $\mathfrak{m}_-$ on morphisms. Let $f : p \rightarrow q$ in $\mathbf{Poly}$. We will define $\mathfrak{m}_f : \mathfrak{m}_p \rightarrow \mathfrak{m}_q$ by inductively defining morphisms $f_{(\alpha)} : p_{(\alpha)} \rightarrow q_{(\alpha)}$ such that for all $\alpha < \beta$ the following diagram commutes:

$$\begin{array}{ccc} p_{(\alpha)} & \xrightarrow{\iota_{(\alpha)}} & p_{(\beta)} \\ f_{(\alpha)} \downarrow & & \downarrow f_{(\beta)} \\ q_{(\alpha)} & \xrightarrow{\iota_{(\alpha)}} & q_{(\beta)} \end{array} \tag{5}$$

Define $f_{(0)} : p_{(0)} \rightarrow q_{(0)}$ to be the identity on y . For a successor ordinal $\alpha + 1$, suppose that we have already defined $f_{(\alpha)} : p_{(\alpha)} \rightarrow q_{(\alpha)}$. Then we define $f_{(\alpha+1)} : p_{(\alpha+1)} \rightarrow q_{(\alpha+1)}$ to be

$$y + p \triangleleft p_{(\alpha)} \xrightarrow{y + f \triangleleft f_{(\alpha)}} y + q \triangleleft q_{(\alpha)}.$$

To show that the diagram in Equation (5) commutes for successor ordinals, it suffices to show that for all α , the diagram with $\beta = \alpha + 1$ commutes. By induction this is immediate by the definitions of $\iota_{(\alpha+1)}$ and $f_{(\alpha+1)}$.

Suppose that α is a limit ordinal and that we have defined $f_{(\alpha')}: p_{(\alpha')} \rightarrow q_{(\alpha')}$ for all $\alpha' < \alpha$. For each $\alpha' < \alpha$ we have maps

$$p_{(\alpha')} \xrightarrow{f_{(\alpha')}} q_{(\alpha')} \xrightarrow{\iota_{(\alpha')}} q_{(\alpha)}.$$

Since the diagram in Equation (5) commutes for all pairs of ordinals less than α , these maps form a cocone into $q_{(\alpha)}$. So, by the universal property of the colimit, $p_{(\alpha)} = \text{colim}_{\alpha' < \alpha} p_{(\alpha')}$, there is an induced map $f_{(\alpha)}: p_{(\alpha)} \rightarrow q_{(\alpha)}$. That the diagram in Equation (5) commutes when β is a limit ordinal follows from the uniqueness of the universal map.

Let κ be such that both p and q are κ -small. Then we define $m_f := f_{(\kappa)}$. We defer the proof that m_f is a map of $\triangleleft$ -monoids to Appendix B.

Proposition 2.10. *For $f: p \rightarrow q$ in **Poly**, the polynomial map $m_f: m_p \rightarrow m_q$ is a map of $\triangleleft$ -monoids.*

It is straightforward to show by transfinite induction that the action of m_- on morphisms is functorial. Therefore, Proposition 2.9 and Proposition 2.10 define a functor $m_-: \mathbf{Poly} \rightarrow \mathbf{Mod}(\mathbf{Poly})$.

Theorem 2.11. *There is an adjunction*

$$\mathbf{Poly} \begin{array}{c} \xrightarrow{m_-} \\ \xleftarrow{U} \end{array} \mathbf{Mod}(\mathbf{Poly}) .$$

Therefore m_p is the **free monad** on the polynomial p and m_- is the **free monad monad**. We defer the proof of Theorem 2.11 and its requisite lemmas to Appendix B.1.

Essentially we have shown that p is κ -small implies that $p_{(\kappa)}$ is *the* free monad on p . Hence we have the following Corollary.

Corollary 2.12. *If $\kappa < \kappa'$ and p is κ -small, then $p_{(\kappa)}$ is isomorphic to $p_{(\kappa')}$.*

Alternatively, we can show directly via transfinite induction that under these hypotheses, the inclusion $\iota_{(\kappa')}: p_{(\kappa)} \rightarrow p_{(\kappa')}$ is an isomorphism. If κ is a limit ordinal and $\kappa' = \kappa + 1$, then its inverse is the map

$$p_{(\kappa+1)} = y + p \triangleleft p_{(\kappa)} = y + p \triangleleft \text{colim}_{\alpha < \kappa} p_{(\alpha)} \xrightarrow{\cong} \text{colim}_{\alpha < \kappa} (y + p \triangleleft p_{(\alpha)}) = \text{colim}_{\alpha < \kappa} p_{(\alpha+1)} = p_{(\kappa)}.$$

The isomorphism follows from Proposition A.1. The remainder of the induction is straightforward.

This corollary implies that the natural map

$$m_p \xrightarrow{\cong} y + p \triangleleft m_p.$$

is an isomorphism. In terms of other words, a p -shaped interview is equivalent to either no interview or a question in $p(1)$ and for every answer, another p -shaped interview.

3 Interactions Between Free Monad and Cofree comonad

3.1 The Cofree Comonad Comonad

It is a beautiful fact that in **Poly** the $\triangleleft$ -comonoids are categories and $\triangleleft$ -comonoid maps are cofunctors. Thus, we use **Cat**[#] to denote the category of $\triangleleft$ -comonoids and their maps. Dual to the construction of the free monad monad in Section 2, here we define the cofree comonad and show that it is right adjoint to the forgetful functor $U: \mathbf{Cat}^\# \rightarrow \mathbf{Poly}$ given by $U(c, \epsilon, \delta) := c$. In Appendix C we prove the statements presented in this Section.

Proposition 3.1. *There is a functor $c_- : \mathbf{Poly} \rightarrow \mathbf{Poly}$ such that c_p has the structure of a $\triangleleft$ -comonoid for each $p : \mathbf{Poly}$,*

$$c_p \rightarrow y \quad \text{and} \quad c_p \rightarrow c_p \triangleleft c_p.$$

For a polynomial functor $q : \mathbf{Poly}$, the positions of the cofree comonad are q -**behavior trees**. These are defined coinductively as a position $Q : q(1)$ and a map from directions of $q[Q]$ to q -behavior trees.

Theorem 3.2. *There is an adjunction*

$$\mathbf{Cat}^\# \begin{array}{c} \xrightarrow{U} \\ \Leftrightarrow \\ \xleftarrow{c_-} \end{array} \mathbf{Poly}.$$

Hence c_q is the **cofree comonad** on the polynomial q and c_- is the **cofree comonad comonad**.

3.2 The Module Structure $m_p \otimes c_q \rightarrow m_{p \otimes q}$

For polynomials p and q consider the map

$$p \otimes c_q \rightarrow p \otimes q \rightarrow m_{p \otimes q}$$

where the first map is induced by the counit of c_q and the second map is the unit of $m_{p \otimes q}$. This composite induces a map of polynomials $p \rightarrow [c_q, m_{p \otimes q}]$. By duoidality of $\otimes$ and $\triangleleft$, the internal hom $[c_q, m_{p \otimes q}]$ is a $\triangleleft$ -monoid as well. Therefore, by the adjunction in Theorem 2.11 the map of polynomials $p \rightarrow [c_q, m_{p \otimes q}]$ induces a map of $\triangleleft$ -monoids $m_p \rightarrow [c_q, m_{p \otimes q}]$, which is equivalent to a polynomial map

$$\Xi_{p,q} : m_p \otimes c_q \rightarrow m_{p \otimes q}.$$

This map is the **free monad-comonad interaction law** described in [4, Section 3.3]. In Appendix D we prove the statements presented in this Section.

Proposition 3.3. *The maps $\Xi_{p,q}$ are natural in p and q .*

Note that c is lax monoidal (see Proposition C.1)

$$c_p \otimes c_q \rightarrow c_{p \otimes q} \quad \text{and} \quad y \rightarrow c_y$$

so in this sense we would say "matter can also take the place of pattern."

Recall from [9] the notion of a module over a monoidal functor.

Theorem 3.4. *There is a left-module over $c_- : (\mathbf{Poly}, \otimes, y) \rightarrow (\mathbf{Poly}, \otimes, y)$ consisting of:*

- **Poly** as a left module category over $(\mathbf{Poly}, \otimes, y)$.
- The functor $m_- : \mathbf{Poly} \rightarrow \mathbf{Poly}$.
- The natural transformation $\Xi : m_- \otimes c_- \Rightarrow m_{-\otimes-}$.

4 Applications

In this section, we will give four applications of the module structure introduced in Section 3. Each consists of:

- A pattern of type p , represented by a map into m_p .
- Matter of type q , represented by a map into c_q .
- A *runs on* map $p \otimes q \rightarrow r$.

Then the composite $m_p \otimes c_q \rightarrow m_{p \otimes q} \rightarrow m_r$ represents the interaction *pattern runs on matter*.

4.1 Interviews Run on People

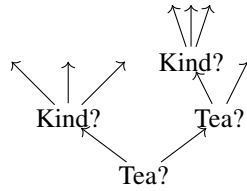
As in Section 2, suppose that the positions of p are questions and the directions are possible answers to each question. For example consider the polynomial, $p = \{\text{Tea?}\}y^{\{\text{yes,no}\}} + \{\text{Kind?}\}y^{\{\text{green,black,herbal}\}}$ which we can view as the corolla forest:



Essentially, p consists of the questions:

- “Do you want tea?”, with possible answers “yes” or “no”.
- “What kind of tea do you like?”, with possible answers “green”, “black”, or “herbal”.

Consider the pattern $y \rightarrow m_p$ which selects the following interview:



The polynomial $[p, y] = \left(\prod_{p:p(1)} p[P] \right) y^{p(1)}$ is the **universal answerer** for the polynomial p . Its positions are a choice of answer for each question in $p(1)$ and its directions are the questions $p(1)$. A person is a map $y \rightarrow c_{[p,y]}$. In other words, a person is a behavior tree in which a node is an answer for each question and a branch is one such question. Each answer may depend on the sequence of questions asked so far. Such a map can always be factored as $y \rightarrow Sy^S \rightarrow c_{[p,y]}$. In other words, a person consists of (1) a dynamical system which outputs answers to p questions and inputs p questions and (2) an initial condition.

Note [4, Section 2.5] refers to $[p, y]$ as the dual of p . We will see in Section 4.4 that it is also interesting to consider generalizations of this duality to $[p, t]$ for any polynomial monad t . Indeed, for any monad t , there is a map

$$m_p \otimes c_{[p,t]} \rightarrow m_t \rightarrow t \quad (6)$$

The evaluation map $p \otimes [p, y] \rightarrow y$ describes how to run a question on a universal answerer. Given an interview and a person we get the composite

$$y \rightarrow m_p \otimes c_{[p,y]} \rightarrow m_{p \otimes [p,y]} \rightarrow m_y = \mathbb{N}y,$$

which sends the single position of y to the number of questions asked when the interview is run on the person. For example, consider Alice who always responds that she does not want tea and she likes herbal tea. Running the tea interview on Alice results in two questions: “Tea?” then “Tea?”. Conversely, consider Bob who at first politely declines tea and likes black tea. Then after the first question, he responds that he actually does want tea and he likes black tea. Running the tea interview on Bob results in three questions: “Tea?”, “Tea?”, then “Kind?”.

4.2 Programs Run on Operating Systems

Consider the following program.

```

def guessing_game(max_guesses, goal):
    if max_guesses==0:
        return False
    guess=read()
    if guess==goal:
        return True
    return guessing_game(max_guesses-1, goal)

```

We represent the argument/return type with the polynomial $r = \sum_{m:\mathbb{N}, g:\mathbb{N}} y^{\text{Bool}}$ where m represents the variable `max_guesses` and g represents the variable `goal`. We represent the effect type that reads in natural numbers with the polynomial $p = y^{\mathbb{N}}$. Finally, we will represent this program with a map $r \rightarrow m_p$ that we define inductively using the decomposition $r = \sum_{m:\mathbb{N}} r_m$ with $r_m = \sum_{g:\mathbb{N}} y^{\text{Bool}}$. We start by defining the following maps:

- Define a map $r_0 \rightarrow y$. Each position of r_0 is sent to the single position of y . The single direction of y is sent to `False : Bool`.
- Define a map $r_{m+1} \rightarrow p \triangleleft (y + r_m)$. On positions, for each position $g : r_{m+1}(1)$, we need a function from $\mathbb{N}$ to the positions of $y + r_m$. In particular, we send g to the map sending $g' : \mathbb{N}$ to y if $g = g'$ and to the position $g : r_m(1)$ otherwise. Then on directions, if $g = g'$, the single direction of y is sent to `True : Bool` and if $g \neq g'$, then we use the identity on directions.

Now we will define maps $r_m \rightarrow m_p$ inductively using the isomorphism $m_p \cong y + p \triangleleft m_p$ and the inclusion $\iota_{(0)} : y \rightarrow m_p$. As a base case, we have the composite $r_0 \rightarrow y \xrightarrow{\iota_{(0)}} m_p$. Given $r_m \rightarrow m_p$, we define

$$r_{m+1} \rightarrow p \triangleleft (y + m_p) \xrightarrow{p \triangleleft (\iota_{(0)}, m_p)} p \triangleleft m_p \rightarrow m_p.$$

A operating system with effects in $[p, y] \cong \mathbb{N}y$ is a map $y \rightarrow c_{[p, y]} \cong (\mathbb{N}y)^{\mathbb{N}}$. It consists of a stream of natural numbers, which are the responses it will give to the `read()` effect.

Using the interaction $\Xi_{p, [p, y]}$ and the evaluation map $p \otimes [p, y] \rightarrow y$, we get the composite

$$r \cong r \otimes y \rightarrow m_p \otimes c_{[p, y]} \rightarrow m_{p \otimes [p, y]} \rightarrow m_y \cong \mathbb{N}y,$$

which expresses how the program runs on the chosen operating system. On positions it maps $(m : \mathbb{N}, g : \mathbb{N})$ to the minimum of m and the number of responses the operating system takes to guess the goal g . On directions, it maps the single direction to `True` if the goal was guessed in at most m guesses and to `False` otherwise.

4.3 Voting Schemes Run on Voters

For a finite set of candidates X , consider the polynomial $p = \sum_{A \subseteq X} y^A$. A position of p is a ballot consisting of some subset of the candidates. For such a position, its directions are possible winners of the ballot. A voting scheme with M voters is a map $p \rightarrow m_{\bigotimes_M p}$ where $\bigotimes_0 p := y$ and $\bigotimes_{M+1} p := p \otimes \bigotimes_M p$. On positions, a subset of candidates $A \subseteq X$ maps to a terminating decision tree in which each node is a personalized ballot given to each of the M voters and each branch corresponds to the tuple of each voter's selection. On directions, each leaf of this decision tree maps to an overall winner.

Exhaustive run-off is a voting scheme in which each voter selects their preference from the A candidates, and then the candidates with the fewest number of votes are eliminated. If only a single candidate remains, then they are elected the winner. Otherwise, another round of voting proceeds with the remaining candidates. This voting scheme can be encoded into a polynomial map $p \rightarrow m_{\bigotimes_M p}$. As

in Section 4.2 we define the map inductively. Consider the decomposition $p = \sum_{n=0}^{\#X} p_n$ where $p_n = \sum_{A \subseteq X, \#A=n} y^A$. Then consider the following maps:

- Note that $p_0 = 1$ is isomorphic to $\bigotimes_M p_0 = \bigotimes_M 1$. There is an inclusion of p_0 into p and hence a map $p_0 \cong \bigotimes_M p_0 \rightarrow \bigotimes_M p$.
- There is a unique map $p_1 \rightarrow y$.
- Define $p_{n+1} \rightarrow (\bigotimes_M p) \triangleleft (\sum_{k=0}^n p_k)$ as follows. On positions $A \subseteq X$ maps to the product $\prod_M A : \prod_M p(1)$ and the map $\prod_M A \rightarrow \sum_{k=0}^n p_k(1)$ defined by

$$(a_1, \dots, a_M) \mapsto A' := A \setminus \arg \min \#_{(a_1, \dots, a_n)}$$

where $\#_{(a_1, \dots, a_n)} : A \rightarrow \mathbb{N}$ counts the number of votes for each candidate and so $\arg \min \#_{(a_1, \dots, a_n)}$ is the set of candidates with the fewest votes. On directions, it is the inclusion $A' \rightarrow A$.

Now, we define $p \rightarrow m_{\bigotimes_M p}$ inductively as follows. As base cases we have the composites

$$p_0 = 1 = \bigotimes_M 1 \rightarrow \bigotimes_M p \rightarrow m_{\bigotimes_M p} \quad \text{and} \quad p_1 \rightarrow y \rightarrow m_{\bigotimes_M p}.$$

Given a maps $p_k \rightarrow m_{\bigotimes_M p}$ for $k < n+1$, we define

$$p_{n+1} \rightarrow \left(\bigotimes_M p \right) \triangleleft \left(\sum_{k=0}^n p_k \right) \rightarrow m_{\bigotimes_M p} \triangleleft \left(\sum_{k=0}^n m_{\bigotimes_M p} \right) \rightarrow m_{\bigotimes_M p} \triangleleft m_{\bigotimes_M p} \xrightarrow{\mu} m_{\bigotimes_M p}.$$

A voter selects a candidate from every subset of candidates. Such a voter is represented by a polynomial map $y \rightarrow [p, y]$. Running M elections on M voters is represented by the composite $(\bigotimes_M p) \otimes (\bigotimes_M [p, y]) \cong \bigotimes_M (p \otimes [p, y]) \rightarrow \bigotimes_M y = y$. Therefore, given M voters, we get a composite

$$p \rightarrow m_{\bigotimes_M p} \otimes \left(\bigotimes_M [p, y] \right) \rightarrow m_{\bigotimes_M p} \otimes c_{\bigotimes_M [p, y]} \rightarrow m_y = \mathbb{N}y.$$

On positions it maps a set of candidates $A \subseteq X$ to the number of run-offs required to elect a candidate. On directions, it maps the single direction of y to the winner.

It is tempting to expect that the maps $p \rightarrow m_{\bigotimes_M p}$ define an operad enriched in the Kleisli category $\mathbf{Poly}_m$ in the sense of [11]. However exhaustive run-off is gerrymander-able meaning that the division of voters into districts can affect the end-result of the election. This observation suggests the following definition of gerrymandering.

Definition 4.1. A voting scheme $p \rightarrow m_{\bigotimes_M p}$ can be **gerrymandered** if and only if it does not extend to an operad enriched in $\mathbf{Poly}_m$. $\diamond$

4.4 Games Run on Players

For a game such as tic-tac-toe, let p be the polynomial whose positions are game states and whose directions are next possible moves. Then we can represent the game play as a position in m_p . In the game tic-tac-toe, a game state is a placement of $\times$'s and $\circ$'s on a 3×3 grid. In other words, it is a map $b : 9 \rightarrow \{\times, \circ, -\}$ where $-$ represents an open grid position. For $m = 1, \dots, 9$, let B_m be the set of valid board states with m open positions. Assuming that $\times$ always plays first, these are

$$B_m := \{b : 9 \rightarrow \{\times, \circ, -\} \mid \#(b^{-1}(-)) = m, \#(b^{-1}(\circ)) \leq \#(b^{-1}(\times)) \leq \#(b^{-1}(\circ)) + 1\}.$$

If there are an odd number of open positions, then it is $\times$'s turn. Otherwise, it is $\circ$'s turn.

Given a board state $b : B_m$, the next possible moves are the open positions $b^{-1}(-)$. Therefore, the polynomial $p_{\times} = \sum_{b: B_1+B_3+\dots+B_9} y^{b^{-1}(-)}$ represents the board states and next possible moves for $\times$. Likewise the polynomial $p_{\circ} = \sum_{b: B_2+B_4+\dots+B_8} y^{b^{-1}(-)}$ represents the board states and next possible moves for $\circ$.

Consider the polynomial $p = p_{\times} \triangleleft (y + p_{\circ})$. This polynomial represents an $\times$ move followed by either game over or an $\circ$ move. There exists a polynomial map $T : y^{\{\times, \circ, -\}} \rightarrow \mathfrak{m}_p$ which selects the decision-tree in $\mathfrak{m}_p$ corresponding to the rules of tic-tac-toe, and maps directions to the winner of a completed game or $-$ if the game is tied. We will define the map T inductively. For $m = 1, \dots, 9$, let $r_m = B_m y^{\{\times, \circ, -\}}$.

- Define $r_1 \rightarrow y$, as follows. Given a board $b : B_1$ with a single open position, send the single direction in y to either the winner or to $-$ if the game is tied.
- Define $r_{m+1} \rightarrow p_{\circ} \triangleleft (y + r_m)$ for odd m , and define $r_{m+1} \rightarrow p_{\times} \triangleleft (y + r_m)$ for even m , as follows. On positions send a board state $b : B_{m+1}$ to itself in $p_{\circ}(1)$ (resp. $p_{\times}(1)$). Then for each valid move $m : b^{-1}(-)$ let $b' : B_m$ be the updated board state. If b' contains a winner, then send m to y and send the single direction of y to the winner. Otherwise, send m to $b' : r_m(1)$ and let the map on directions be the identity.

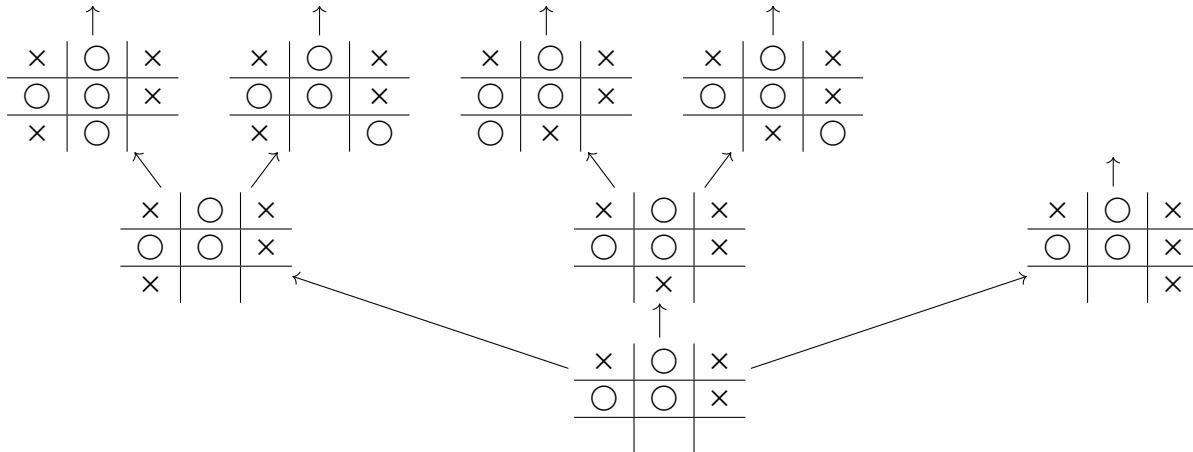
Using these maps, we can inductively define maps $r_m \rightarrow \mathfrak{m}_p$ for odd m as follows. As a base case we have the composite $r_1 \rightarrow y \rightarrow \mathfrak{m}_p$. Given a map $r_m \rightarrow \mathfrak{m}_p$, we have the composite

$$\begin{aligned} r_{m+2} &\rightarrow p_{\times} \triangleleft (y + p_{\circ} \triangleleft (y + r_m)) \rightarrow p_{\times} \triangleleft (y + p_{\circ} \triangleleft (y + \mathfrak{m}_p)) \rightarrow p_{\times} \triangleleft (y + p_{\circ} \triangleleft \mathfrak{m}_p) \\ &\rightarrow p_{\times} \triangleleft (y + p_{\circ}) \triangleleft \mathfrak{m}_p = p \triangleleft \mathfrak{m}_p \rightarrow \mathfrak{m}_p. \end{aligned}$$

The second to last map is induced by the composite

$$y + p_{\circ} \triangleleft \mathfrak{m}_p = y \triangleleft y + p_{\circ} \triangleleft \mathfrak{m}_p \rightarrow y \triangleleft \mathfrak{m}_p + p_{\circ} \triangleleft \mathfrak{m}_p \rightarrow (y + p_{\circ}) \triangleleft \mathfrak{m}_p.$$

Below is the image of a board state in $r_3(1)$ under the map $r_3 \rightarrow \mathfrak{m}_p$. From left to right the directions are sent to $\circ$, $-$, $-$, $-$, and $\times$, as these are the win/loss/tie results of the games as shown.



Define T to be the map $r_9 = y^{\{\times, \circ, -\}} \rightarrow \mathfrak{m}_p$.

Let (t, η, μ) be a $\triangleleft$ -monad. A polynomial map $\varphi_{\times} : y \rightarrow [p_{\times}, t]$ represents an $\times$ player and a polynomial map $\varphi_{\circ} : y \rightarrow [p_{\circ}, t]$ represents an $\circ$ player. If t is the trivial monad y , then for each board

state, the player's next move is deterministic. If t is the lottery monad $\text{lott} = \sum_{M:\mathbb{N}} \sum_{p:\Delta M} y^M$, then the players' next moves are stochastic.

By duoidality we have

$$p \otimes [p_{\times}, t] \otimes [p_{\circ}, t] \rightarrow p \otimes [p_{\times}, t] \otimes [y + p_{\circ}, t] \rightarrow (p_{\times} \otimes [p_{\times}, t]) \triangleleft ((y + p_{\circ}) \otimes [y + p_{\circ}, t]) \rightarrow t \triangleleft t \rightarrow t$$

where the first map is induced by the monoidal unit of t . Given an $\times$ player and an $\circ$ player, we have a map $y^{\{\times, \circ, -\}} \cong y^{\{\times, \circ, -\}} \otimes y \otimes y \xrightarrow{T \otimes ! \otimes !} m_p \otimes c_y \otimes c_y$. Then, the game play is represented by the composite

$$y^{\{\times, \circ, -\}} \rightarrow m_p \otimes c_y \otimes c_y \xrightarrow{m_p \otimes c_{\varphi_{\times}} \otimes c_{\varphi_{\circ}}} m_p \otimes c_{[p_{\times}, t]} \otimes c_{[p_{\circ}, t]} \rightarrow m_{p \otimes [p_{\times}, t] \otimes [p_{\circ}, t]} \rightarrow m_t \rightarrow t \quad (7)$$

which maps directions in t to winners of completed games.

We can promote this setup to players which *learn*, in other words players whose strategy dynamically changes after each completed game. A dynamic $\times$ player consists of a set of states $S_{\times}$ and a polynomial map $S_{\times} y^{S_{\times}} \rightarrow y^{\{\times, \circ, -\}} \otimes c_{[p_{\times}, t]}$. Likewise for a dynamic $\circ$ player. Such a player consists of the following:

- For each state, a behavior tree describing the player's strategy.
- For a winner or tied game and each finite path of the behavior tree, a new state.

Creating maps like φ is the subject of reinforcement learning [12]. As a typical such algorithm, take $S_{\times}$ to be the set of functions that assign a score in $\mathbb{N}$ to each move (direction in $p_{\times}$). Let $S_{\times} y^{S_{\times}} \rightarrow y^{\{\times, \circ, -\}} \otimes c_{[p_{\times}, \text{lott}]}$ be defined as follows:

- On positions, it takes a score for each move and assigns a (stochastic) strategy which selects moves based on their relative scores.
- On directions, note that a direction of $c_{[p_{\times}, \text{lott}]}$ contains a finite number of moves played by $\times$ in the game. If the winner is $\times$, then we add 1 to the score for each played move. Otherwise, we keep the original scores.

Using the copy-on-directions map $y^{\{\times, \circ, -\}} \otimes y^{\{\times, \circ, -\}} \rightarrow y^{\{\times, \circ, -\}}$, we have the composite

$$\begin{aligned} S_{\times} y^{S_{\times}} \otimes S_{\circ} y^{S_{\circ}} &\rightarrow y^{\{\times, \circ, -\}} \otimes y^{\{\times, \circ, -\}} \otimes c_{[p_{\times}, t]} \otimes c_{[p_{\circ}, t]} \\ &\rightarrow y^{\{\times, \circ, -\}} \otimes c_{[p_{\times}, t]} \otimes c_{[p_{\circ}, t]} \xrightarrow{T} m_p \otimes c_{[p_{\times}, t]} \otimes c_{[p_{\circ}, t]} \rightarrow t. \end{aligned}$$

This map takes a state for each player, runs the game using each player's strategy, and returns an updated state for each player based on the winner and the player's strategy.

5 Conclusion

In this paper, we constructed the free monad m_p and cofree comonad c_q on arbitrary polynomial functors $p, q : \mathbf{Poly}$, and defined a module structure $m_p \otimes c_q \rightarrow m_{p \otimes q}$. We also gave a series of examples to explain how this models the intuition "pattern runs on matter."

From here, it is not hard to show that m and c respectively extend to a monad and a comonad on $\mathbf{Org}$, the double category which serves as the base of enrichment for dynamic categorical structures for deep learning, prediction markets, etc., as defined in [11]. We (or others) may show in future work that for any polynomial monad t , there is a functor $[-, t] : \mathbf{Org}_{\text{m}}^{\text{op}} \rightarrow \mathbf{Org}^c$ from opposite of the m -Kleisli category to the c -coKleisli category. The latter offers the ability for different machines to operate at different rates in

wiring diagrams, and the former offers the ability to call multiple subprocesses before returning, though we have not found compelling examples of either; this again is future work.

The constructions in this paper should generalize straightforwardly to free monads and cofree comonads for familial functors between copresheaf categories, as in [6, 13]. One should check that there is again a module structure of the same form in that setting.

References

- [1] Samson Abramsky, Dov M Gabbay & Thomas SE Maibaum (2001): *Handbook of Logic in Computer Science: Volume 5. Algebraic and Logical Structures*. OUP Oxford, doi:10.1093/oso/9780198537816.001.0001.
- [2] Jiri Adamek & Jiri Rosicky: *Locally presentable and accessible categories*. London Mathematical Society Lecture Note Series 189, Cambridge University Press, doi:10.1017/CBO9780511600579.
- [3] Nicola Gambino & Joachim Kock (2012): *Polynomial functors and polynomial monads*. *Mathematical Proceedings of the Cambridge Philosophical Society* 154(1), p. 153–192, doi:10.1017/S0305004112000394.
- [4] Shin-ya Katsumata, Exequiel Rivas & Tarmo Uustalu (2020): *Interaction Laws of Monads and Comonads*. doi:10.1145/3373718.3394808.
- [5] G Max Kelly (1980): *A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on*. *Bulletin of the Australian Mathematical Society* 22(1), pp. 1–83, doi:10.1017/S0004972700006353.
- [6] Owen Lynch, Brandon T. Shapiro & David I. Spivak (2023): *All Concepts are $\mathbf{Cat}^{\#}$* , doi:10.48550/arXiv.2305.02571.
- [7] Nelson Niu & David I. Spivak (2022): *Polynomial functors: A mathematical theory of interaction*. doi:10.48550/arXiv.2312.00990. *In preparation*.
- [8] nLab authors (2024): *Exact square*. <https://ncatlab.org/nlab/show/exact+square>. Revision 17.
- [9] nLab authors (2024): *Module over a monoidal functor*. <https://ncatlab.org/nlab/show/module+over+a+monoidal+functor>. Revision 4.
- [10] nLab authors (2024): *Transfinite construction of free algebras*. <https://ncatlab.org/nlab/show/transfinite+construction+of+free+algebras>. Revision 13.
- [11] Brandon T. Shapiro & David I. Spivak (2022): *Dynamic categories, dynamic operads: From deep learning to prediction markets*. *Electronic Proceedings in Theoretical Computer Science*, doi:10.4204/EPTCS.380.11.
- [12] Richard S Sutton & Andrew G Barto (2018): *Reinforcement learning: An introduction*. MIT press.
- [13] Mark Weber: *Familial 2-Functors and Parametric Right Adjoints*. *Theory and Applications of Categories* 18(22), pp. Paper No. 22, 665–732.

A Distributing Colimits over $\triangleleft$

In general, colimits do not distribute over the substitution product $\triangleleft$. However, in this section we give hypotheses under which the natural maps

$$\operatorname{colim}_{j \in \mathcal{J}} (p \triangleleft q_j) \rightarrow p \triangleleft (\operatorname{colim}_{j \in \mathcal{J}} q_j) \quad \text{and} \quad \operatorname{colim}_{j \in \mathcal{J}} (p_j \triangleleft q) \rightarrow (\operatorname{colim}_{j \in \mathcal{J}} p_j) \triangleleft q$$

are in fact isomorphisms.

A.1 Right Distribution

We begin with hypotheses under which colimits right distribute over $\triangleleft$.

Proposition A.1. *If a polynomial p is κ -small, then for all κ -filtered categories $\mathcal{J}$ and diagrams $q : \mathcal{J} \rightarrow \mathbf{Poly}$, the natural map*

$$\operatorname{colim}_{j: \mathcal{J}} (p \triangleleft q_j) \rightarrow p \triangleleft \left(\operatorname{colim}_{j: \mathcal{J}} q_j \right)$$

is an isomorphism.

Proof. That this map is an isomorphism on positions follows directly from the definition of κ -small. That this map is an isomorphism on directions follows from the definition of colimits in **Poly** and the fact that connected limits preserve coproducts. $\square$

A.2 Left Distribution

Next we consider hypotheses under which colimits left distribute over $\triangleleft$. Let **Poly**_{Cart} be subcategory of **Poly** consisting of cartesian maps. If ϕ is cartesian, then so is $p \triangleleft \phi$ and $p + \phi$ for any $p : \mathbf{Poly}$.

First we give a series of Lemmas which allow us to prove Proposition A.6.

Lemma A.2. *A polynomial map $p \rightarrow q$ is an isomorphism if and only if for all sets X , the induced function $p \triangleleft X \rightarrow q \triangleleft X$ is a bijection.*

Proof. This is just a restatement of the fact that $\operatorname{Ext} : \mathbf{Poly} \rightarrow \mathbf{Set}^{\mathbf{Set}}$ is fully faithful; indeed, $\operatorname{Ext}(p)(X) \cong p \triangleleft X$. $\square$

The coproduct completion $\Sigma \mathcal{C}$ of a category $\mathcal{C}$ has as objects pairs (S, C) where $S : \mathbf{Set} \rightarrow \mathcal{C}$ and $C : \operatorname{El} S \rightarrow \mathcal{C}$ is a discrete diagram, and it has as morphisms lax triangles

$$\begin{array}{ccc} S_1 & \xrightarrow{f} & S_2 \\ & \searrow \scriptstyle C_1 & \swarrow \scriptstyle C_2 \\ & \xrightarrow{f^b} & \\ & \scriptstyle \mathcal{C} & \end{array}$$

A diagram $\mathcal{J} \rightarrow \Sigma \mathcal{C}$ consists of a pair (S, C) , where $S : \mathcal{J} \rightarrow \mathbf{Set}$ is a diagram and $C : \operatorname{El} S \rightarrow \mathcal{C}$ is a functor from the category of elements of S to $\mathcal{C}$.

Lemma A.3. *Let $\mathcal{C}$ be a category and let $\Sigma \mathcal{C}$ be its coproduct completion. If $\mathcal{C}$ has $\mathcal{J}$ -shaped colimits, then so does $\Sigma \mathcal{C}$, and they are computed as follows.*

Given a diagram $(S, C) : \mathcal{J} \rightarrow \Sigma \mathcal{C}$, consider the Kan extension

$$\begin{array}{ccc} \operatorname{El} S & \xrightarrow{C} & \mathcal{C} \\ \downarrow & \Downarrow & \uparrow \\ \pi_0(\operatorname{El} S) & \xrightarrow{\operatorname{Lan}} & \end{array}$$

where $\pi_0 : \mathbf{Cat} \rightarrow \mathbf{Set}$ is the connected components reflection. Then the colimit is given by

$$\operatorname{colim}(S, C) \cong (\pi_0(\operatorname{El} S), \operatorname{Lan}). \quad (8)$$

For each element $s_0 : \pi_0(\operatorname{El} S)$, its image object $\operatorname{Lan}(s_0) : \operatorname{Ob} \mathcal{C}$ is given by the colimit over all $(i, s_i) \mapsto s$ of $C(s_i)$.

Proof. First consider the case $\mathcal{C} = 1$, so that $\Sigma\mathcal{C} \cong \mathbf{Set}$. We need to check that for any $S : \mathcal{I} \rightarrow \mathbf{Set}$, there is a bijection

$$\mathrm{colim} S \cong \pi_0(\mathrm{El} S). \quad (9)$$

This follows from the fact that the inclusion of diagrams of sets into diagrams of categories is a left adjoint, the Grothendieck construction taking diagrams of categories to categories is a left adjoint, and π_0 taking categories to sets is a left adjoint.

Given $T : \mathbf{Set}$, a discrete diagram $D : T \rightarrow \mathcal{C}$, and diagrams $(S_i, C_i) \rightarrow (T, D)$, coherently over $i : \mathcal{I}$, one obtains a diagram as left, which factors uniquely as right:

$$\begin{array}{ccc} \mathrm{El} S & \longrightarrow & T \\ \downarrow c & \Rightarrow & \downarrow D \\ & \mathcal{C} & \end{array} \qquad \begin{array}{ccccc} \mathrm{El} S & \longrightarrow & \pi_0(\mathrm{El} S) & \longrightarrow & T \\ \downarrow c & \Rightarrow & \downarrow \mathrm{Lan} & \Rightarrow & \downarrow D \\ & \mathcal{C} & & & \end{array}$$

This gives (8). For the last statement, note that since $1 \rightarrow \pi_0(\mathrm{El} S)$ is a map of sets, the pullback of categories

$$\begin{array}{ccc} \bullet & \longrightarrow & \mathrm{El} S \\ \downarrow & \lrcorner & \downarrow \\ 1 & \xrightarrow{s_0} & \pi_0(\mathrm{El} S) \end{array}$$

is also a comma square. Hence it is exact in the sense of [8], completing the proof. $\square$

Lemma A.3 provides a formula for computing colimits in $\mathbf{Poly}$, since $\mathbf{Poly} \cong \Sigma \mathbf{Set}^{\mathrm{op}}$ is the coproduct completion of $\mathbf{Set}^{\mathrm{op}}$. Namely, the position-set of the colimit of $p : \mathcal{I} \rightarrow \mathbf{Poly}$ is given by the colimit $\mathrm{colim}_i p_i(1)$ of the position-sets, and the directions at a position $P : \mathrm{colim}_i p_i(1)$ are given by the limit (taken in $\mathbf{Set}$, i.e. the colimit taken in $\mathbf{Set}^{\mathrm{op}}$) of the associated connected diagram $\mathcal{E}_P$ of direction sets.

$$\begin{array}{ccccc} \mathcal{E}_P & \longrightarrow & \mathrm{El} p(1) & \xrightarrow{p[-]} & \mathbf{Set}^{\mathrm{op}} \\ \downarrow & \lrcorner & \downarrow & \Downarrow & \uparrow \\ 1 & \xrightarrow{p} & \mathrm{colim}_i p_i(1) & \xrightarrow{\mathrm{Lan}} & \end{array}$$

Proposition A.4. *The inclusion $\mathbf{Poly}_{\mathrm{Cart}} \rightarrow \mathbf{Poly}$ reflects all colimits.*

Proof. Suppose given a category I , a functor $p : I \rightarrow \mathbf{Poly}_{\mathrm{Cart}}$, a polynomial $\bar{p} : \mathbf{Poly}$, and cartesian maps $\varphi_i : p_i \rightarrow \bar{p}$ forming a cone. Suppose further that in $\mathbf{Poly}$ it is a colimit cone. To see that it is a colimit cone in $\mathbf{Poly}_{\mathrm{Cart}}$ we need only check that for any q and cone $\psi_i : p_i \rightarrow q$ in $\mathbf{Poly}_{\mathrm{Cart}}$, the induced map $\psi : \bar{p} \rightarrow q$ in $\mathbf{Poly}$ is in fact in $\mathbf{Poly}_{\mathrm{Cart}}$, i.e. for any position $P : \bar{p}(1)$, the induced function $\psi_P^\# : q[\varphi(P)] \rightarrow \bar{p}[P]$ is bijective.

By Lemma A.3, the position-set of a colimit is the colimit of the position-sets for any diagram of polynomials, meaning there exists some $i : I$ and $P_i : p_i(1)$ representing P , i.e. with $\varphi_i(P_i) = P$. Since $p_i \rightarrow \bar{p}$ is cartesian, we have a commuting triangle of functions $q[\varphi(P)] \rightarrow \bar{p}[P] \rightarrow p_i[P_i]$ of which two are bijections; it follows that the required one is as well. $\square$

Proposition A.5. *The inclusion $\mathbf{Poly}_{\mathbf{Cart}} \rightarrow \mathbf{Poly}$ creates coproducts and filtered colimits. Moreover, the composite*

$$\mathbf{Poly}_{\mathbf{Cart}} \rightarrow \mathbf{Poly} \xrightarrow{\text{Ext}} \mathbf{Set}^{\mathbf{Set}}$$

creates coproducts and filtered colimits.

Proof. The functor $\text{Ext}: \mathbf{Poly} \rightarrow \mathbf{Set}^{\mathbf{Set}}$ is fully faithful and hence reflects all colimits; by Proposition A.4, $\mathbf{Poly}_{\mathbf{Cart}} \rightarrow \mathbf{Poly}$ does too. It suffices to show that $\mathbf{Poly}_{\mathbf{Cart}}$ has coproducts and filtered colimits, because and that the two functors $\mathbf{Poly}_{\mathbf{Cart}} \rightarrow \mathbf{Poly}$ and $\mathbf{Poly}_{\mathbf{Cart}} \rightarrow \mathbf{Set}^{\mathbf{Set}}$ preserve them.

Coproduct inclusions of polynomials are cartesian, so the coproduct cone exists in $\mathbf{Poly}_{\mathbf{Cart}}$, and hence it is a coproduct. Since Ext preserves coproducts, this concludes the case for coproducts.

By [2], a category has (resp. a functor preserves) all filtered colimits iff it has (resp. preserves) all directed colimits, so let $\mathcal{I}$ be a directed category and $p: \mathcal{I} \rightarrow \mathbf{Poly}_{\mathbf{Cart}}$ a directed sequence of polynomials and cartesian maps. Then the induced map $\mathcal{I} \rightarrow \mathbf{Poly}$ has a colimit, say $\bar{p}$, and it is easy to check that all the structure maps $p_i \rightarrow \bar{p}$ are cartesian. Hence $\bar{p} \cong \text{colim}_i p_i$ is a colimit in $\mathbf{Poly}_{\mathbf{Cart}}$, and it is preserved by the inclusion.

It remains to check that $\bar{p}$ is a colimit in $\mathbf{Set}^{\mathbf{Set}}$. That is, we need to check that for any $X: \mathbf{Set}$, the function

$$\sum_{P: \pi_0(\text{El } p)} \text{colim}_{e: \mathcal{E}_P} X^{p[e]} \longrightarrow \sum_{P: \pi_0(\text{El } p)} X^{\text{lim}_{e: \mathcal{E}_P} p[e]}$$

is a bijection. Choose any P ; it suffices to show that $\text{colim}_{e: \mathcal{E}_P} X^{p[e]} \rightarrow X^{\text{lim}_{e: \mathcal{E}_P} p[e]}$ is a bijection. But $\mathcal{E}_P$ is filtered and for any $e \rightarrow e'$ in $\mathcal{E}_P$, the map $p[e'] \rightarrow p[e]$ is a bijection. Hence we can pick any object $e': \mathcal{E}_P$, replace all p_e with $p_{e'}$, and have an isomorphic diagram. Now the colimit and limit are both constant, meaning that both sides of the desired map are isomorphic to $X^{p[e']}$, and hence to each other. $\square$

Proposition A.6. *For any polynomial q , filtered category $\mathcal{I}$, and diagram $p: \mathcal{I} \rightarrow \mathbf{Poly}_{\mathbf{Cart}}$, the natural map*

$$\text{colim}_{i: \mathcal{I}} (p_i \triangleleft q) \rightarrow (\text{colim}_{i: \mathcal{I}} p_i) \triangleleft q$$

is an isomorphism in $\mathbf{Poly}$.

Proof. By Lemma A.2 it suffices to show that the function $\text{colim}_{i: \mathcal{I}} (p_i \triangleleft q) \triangleleft X \rightarrow (\text{colim}_{i: \mathcal{I}} p_i) \triangleleft q \triangleleft X$ is a bijection for any $X: \mathbf{Set}$. This is just two uses of Proposition A.5:

$$\text{colim}_{i: \mathcal{I}} (p_i \triangleleft q) \triangleleft X \cong \text{colim}_{i: \mathcal{I}} (p_i \triangleleft q \triangleleft X) \cong (\text{colim}_{i: \mathcal{I}} p_i) \triangleleft q \triangleleft X. \quad \square$$

B Proofs for the Free Monad Monad

Lemma 2.8. *For ordinals α, β, γ , the following diagram commutes:*

$$\begin{array}{ccc} p_{(\alpha)} \triangleleft p_{(\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{p_{(\alpha)} \triangleleft \mu_{(\beta, \gamma)}} & p_{(\alpha)} \triangleleft p_{(\beta + \gamma)} \\ \mu_{(\alpha, \beta)} \triangleleft p_{(\gamma)} \downarrow & & \downarrow \mu_{(\alpha, \beta + \gamma)} \\ p_{(\alpha + \beta)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha + \beta, \gamma)}} & p_{(\alpha + \beta + \gamma)} \end{array}$$

Proof. We prove this lemma by transfinite induction on α . First, for $\alpha = 0$ the result is immediate.

Next, suppose that the result holds for α , and we will show that it holds for the successor ordinal $\alpha + 1$. By definition of $p_{(\alpha+1)}$, it suffices to show that the outer square in the following diagram commutes.

$$\begin{array}{ccc}
 p_{(\beta)} \triangleleft p_{(\gamma)} + p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\beta,\gamma)} + p \triangleleft p_{(\alpha)} \triangleleft \mu_{(\beta,\gamma)}} & p_{(\beta+\gamma)} + p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta+\gamma)} \\
 \downarrow p_{(\beta)} \triangleleft p_{(\gamma)} + p \triangleleft \mu_{(\alpha,\beta)} \triangleleft p_{(\gamma)} & & \downarrow p_{(\beta+\gamma)} + p \triangleleft \mu_{(\alpha,\beta+\gamma)} \\
 p_{(\beta)} \triangleleft p_{(\gamma)} + p \triangleleft p_{(\alpha+\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\beta,\gamma)} + p_{(\alpha)} \triangleleft \mu_{(\alpha+\beta,\gamma)}} & p_{(\beta+\gamma)} + p_{(\alpha+\beta+\gamma)} \\
 \downarrow (\iota_{(\beta)} \triangleleft p_{(\gamma)}, \lambda_{(\alpha+\beta)} \triangleleft p_{(\gamma)}) & & \downarrow (\iota_{(\beta+\gamma)}, \lambda_{(\alpha+\beta+\gamma)}) \\
 p_{(\alpha+\beta+1)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha+\beta+1,\gamma)}} & p_{(\alpha+\beta+\gamma+1)}
 \end{array}$$

The top square commutes by the induction hypothesis. To check that the bottom squares commute, it suffices to show that the following diagrams commute as well.

$$\begin{array}{ccc}
 p_{(\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\beta,\gamma)}} & p_{(\beta+\gamma)} \\
 \downarrow \iota_{(\beta)} \triangleleft p_{(\gamma)} & & \downarrow \iota_{(\beta+\gamma)} \\
 p_{(\alpha+\beta+1)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha+\beta+1,\gamma)}} & p_{(\alpha+\beta+\gamma+1)}
 \end{array}
 \quad \text{and} \quad
 \begin{array}{ccc}
 p \triangleleft p_{(\alpha+\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{p \triangleleft \mu_{(\alpha+\beta,\gamma)}} & p \triangleleft p_{(\alpha+\beta+\gamma)} \\
 \downarrow \lambda_{(\alpha+\beta)} \triangleleft p_{(\gamma)} & & \downarrow \lambda_{(\alpha+\beta+\gamma)} \\
 p_{(\alpha+\beta+1)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha+\beta+1,\gamma)}} & p_{(\alpha+\beta+\gamma+1)}
 \end{array}$$

The diagram on the left commutes by Equation (4). The diagram on the right commutes by the inductive definition of $\mu_{(\alpha+\beta+1,\gamma)}$.

Finally, suppose that α is a limit ordinal and suppose that the diagram commutes for all $\alpha' < \alpha$. Recall that $p_{(\alpha)} = \text{colim}_{\alpha' < \alpha} p_{(\alpha')}$. So it suffices to show that the outer diagram in the following diagram commutes.

$$\begin{array}{ccccc}
 p_{(\alpha')} \triangleleft p_{(\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{p_{(\alpha')} \triangleleft \mu_{(\beta,\gamma)}} & p_{(\alpha')} \triangleleft p_{(\beta+\gamma)} & \longrightarrow & (\text{colim}_{\alpha' < \alpha} p_{(\alpha')}) \triangleleft p_{(\beta+\gamma)} \\
 \downarrow \mu_{(\alpha',\beta)} \triangleleft p_{(\gamma)} & & \downarrow \mu_{(\alpha',\beta+\gamma)} & & \downarrow \mu_{(\alpha,\beta+\gamma)} \\
 p_{(\alpha'+\beta)} \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha'+\beta,\gamma)}} & p_{(\alpha'+\beta+\gamma)} & & \\
 \downarrow & & \searrow & & \downarrow \\
 (\text{colim}_{\alpha' < \alpha} p_{(\alpha'+\beta)}) \triangleleft p_{(\gamma)} & \xrightarrow{\mu_{(\alpha+\beta,\gamma)}} & \text{colim}_{\alpha' < \alpha} p_{(\alpha'+\beta+\gamma)} & &
 \end{array}$$

The upper left square commutes by the induction hypothesis and the remaining diagrams commute by the definition of μ for limit ordinals. $\square$

Proposition 2.10. For $f : p \rightarrow q$ in **Poly**, the polynomial map $m_f : m_p \rightarrow m_q$ is a map of $\triangleleft$ -monoids.

Proof. To show that m_f respects the identity it suffices to show that the outer diagram in the following commutes:

$$\begin{array}{ccccc} y & \xrightarrow{y} & p_{(0)} & \xrightarrow{\iota_{(0)}} & m_p \\ & \searrow y & \downarrow f_{(0)} & & \downarrow m_f \\ & & q_{(0)} & \xrightarrow{\iota_{(0)}} & m_q \end{array}$$

The left triangle commutes because each map is the identity on y . The square commutes as it is the diagram in Equation (5).

To show that m_f respects multiplication, it suffices to show that the following diagram commutes for all α and β .

$$\begin{array}{ccc} p_{(\alpha)} \triangleleft p_{(\beta)} & \xrightarrow{\mu_{(\alpha,\beta)}} & p_{(\alpha+\beta)} \\ f_{(\alpha)} \triangleleft f_{(\beta)} \downarrow & & \downarrow f_{(\alpha+\beta)} \\ q_{(\alpha)} \triangleleft q_{(\beta)} & \xrightarrow{\mu_{(\alpha,\beta)}} & q_{(\alpha+\beta)} \end{array}$$

We show this by induction on α . For $\alpha = 0$ this diagram commutes because the horizontal maps are the identity and the vertical maps are $f_{(\beta)}$. Suppose that the diagram commutes for α . Then for the successor ordinal $\alpha + 1$, following the definitions of $\mu_{(\alpha+1,\beta)}$ and $f_{(\alpha+1,\beta)}$ we want to show that the following diagram commutes.

$$\begin{array}{ccc} p_{(\beta)} + p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta)} & \xrightarrow{(\iota_{(\beta)}, (p \triangleleft \mu_{(\alpha,\beta)}) \circ \lambda_{(\alpha+\beta)})} & p_{(\alpha+\beta+1)} \\ f_{(\beta)} + f \triangleleft f_{(\alpha)} \triangleleft f_{(\beta)} \downarrow & & \downarrow f_{(\alpha+\beta+1)} \\ q_{(\beta)} + q \triangleleft q_{(\alpha)} \triangleleft q_{(\beta)} & \xrightarrow{(\iota_{(\beta)}, (p \triangleleft \mu_{(\alpha,\beta)}) \circ \lambda_{(\alpha+\beta)})} & q_{(\alpha+\beta+1)} \end{array}$$

The diagram commutes on the first term of the coproduct because the diagram in Equation (5) commutes with the pair $\beta < \alpha + \beta$. To show that the diagram commutes on the second term, it suffices to show that the outer diagram in the following commutes.

$$\begin{array}{ccccc} p \triangleleft p_{(\alpha)} \triangleleft p_{(\beta)} & \xrightarrow{p \triangleleft \mu_{(\alpha,\beta)}} & p \triangleleft p_{(\alpha+\beta)} & \xrightarrow{\lambda_{(\alpha+\beta)}} & p_{(\alpha+\beta+1)} \\ f \triangleleft f_{(\alpha)} \triangleleft f_{(\beta)} \downarrow & & \downarrow f \triangleleft f_{(\alpha+\beta)} & & \downarrow f_{(\alpha+\beta+1)} \\ q \triangleleft q_{(\alpha)} \triangleleft q_{(\beta)} & \xrightarrow{q \triangleleft \mu_{(\alpha,\beta)}} & q \triangleleft q_{(\alpha+\beta)} & \xrightarrow{\lambda_{(\alpha+\beta)}} & q_{(\alpha+\beta+1)} \end{array}$$

The left-hand square commutes by the induction hypothesis and the right-hand square commutes by definition of $f_{(\alpha+\beta+1)}$. $\square$

B.1 Proof of Theorem 2.11

The trickiest part of proving the adjunction in Theorem 2.11 is defining the co-unit so we do it separately in the following series of Lemmas.

Lemma B.1. *Let (q, η_q, μ_q) be a $\triangleleft$ -monoid. For each ordinal α , there exist a cocone of maps*

$$\epsilon_{(\alpha)} : q_{(\alpha)} \rightarrow q.$$

Proof. We will define the cocone of maps $\epsilon_{(\alpha)}$ inductively. First, define $\epsilon_{(0)}: q_{(0)} = y \rightarrow q$ to be the unit η_q .

Next, suppose that we have a cocone of maps $\epsilon_{(\alpha')}: q_{(\alpha')} \rightarrow q$ for all $\alpha' < \alpha$. If α is a limit ordinal then we define $\epsilon_{(\alpha)}: q_{(\alpha)} \rightarrow q$ to be the universal map induced by the cocone. If $\alpha + 1$ is a successor ordinal, then we define $\epsilon_{(\alpha+1)}$ to be the composite

$$q_{(\alpha+1)} = y + q \triangleleft q_{(\alpha)} \xrightarrow{\eta_q + q \triangleleft \epsilon_{(\alpha)}} q + q \triangleleft q \xrightarrow{(1, \mu_q)} q.$$

To show that these maps form a cocone, it suffices to show that the following diagram commutes for all $\alpha' < \alpha$.

$$\begin{array}{ccc} q_{(\alpha')} & & \\ \downarrow \iota_{(\alpha')} & \searrow \epsilon_{(\alpha')} & \\ q_{(\alpha)} & \xrightarrow{\epsilon_{(\alpha)}} & q \end{array}$$

If α is a limit ordinal, then this is immediate. Otherwise, α is a successor ordinal, say $\alpha = \beta + 1$. It suffices to show that the following diagram commutes.

$$\begin{array}{ccc} q_{(\beta)} & & \\ \downarrow \iota_{(\beta)} & \searrow \epsilon_{(\beta)} & \\ q_{(\beta+1)} & \xrightarrow{\epsilon_{(\beta+1)}} & q \end{array}$$

Consider the following three cases.

- Suppose that $\beta = 0$, then it is immediate that the desired diagram commutes as we see below.

$$\begin{array}{ccc} q_{(0)} = y & \xrightarrow{\eta_q} & q \\ \downarrow \iota_{(0)} & & \searrow \\ q_{(1)} = y + q \triangleleft y & \xrightarrow{\eta_q + \mu_q \circ (q \triangleleft \eta_q)} & q + q \xrightarrow{(1,1)} q \end{array}$$

- Next suppose that β is a successor ordinal. In particular $\beta = \gamma + 1$. To show that

$$\begin{array}{ccc} q_{(\gamma+1)} & & \\ \downarrow \iota_{(\gamma+1)} & \searrow \epsilon_{(\gamma+1)} & \\ q_{(\gamma+2)} & \xrightarrow{\epsilon_{(\gamma+2)}} & q \end{array}$$

commutes, it suffices to show that

$$\begin{array}{ccc} y + q \triangleleft q_{(\gamma)} & & \\ \downarrow y + q \triangleleft \iota_{(\gamma)} & \searrow \eta_q + q \triangleleft \epsilon_{(\gamma)} & \\ y + q \triangleleft q_{(\gamma+1)} & \xrightarrow{\eta_q + q \triangleleft \epsilon_{(\gamma+1)}} & q + q \triangleleft q \end{array}$$

commutes. This is immediate from the induction hypothesis.

- Finally, suppose that β is a limit ordinal. Then we want to show that the outer diagram in the following commutes.

$$\begin{array}{ccccc}
 q_{(\beta')} & & & & \\
 \downarrow \iota_{(\beta')} & \searrow \epsilon_{(\beta')} & & & \\
 q_{(\beta'+1)} = y + q \triangleleft q_{(\beta')} & & & & \\
 \downarrow y + q \triangleleft \iota_{(\beta')} & \searrow \eta_q + q \triangleleft \epsilon_{(\beta')} & \searrow \epsilon_{(\beta'+1)} & & \\
 y + q \triangleleft q_{(\beta)} & \xrightarrow{\eta_q + q \triangleleft \epsilon_{(\beta)}} & q + q \triangleleft q & \xrightarrow{(1, \mu_q)} & q
 \end{array}$$

The top and bottom triangles commute by the induction hypothesis and the middle triangle commutes by definition of $\epsilon_{(\beta'+1)}$.

□

Lemma B.2. For all ordinals α, β the following diagram commutes.

$$\begin{array}{ccc}
 q_{(\alpha)} \triangleleft q_{(\beta)} & \xrightarrow{\mu_{(\alpha, \beta)}} & q_{(\alpha + \beta)} \\
 \downarrow \epsilon_{(\alpha)} \triangleleft \epsilon_{(\beta)} & & \downarrow \epsilon_{(\alpha + \beta)} \\
 q \triangleleft q & \xrightarrow{\mu_q} & q
 \end{array}$$

Proof. We induct on α . Suppose that for all $\alpha' < \alpha$ the diagram commutes. Consider the following cases.

- For $\alpha = 0$, we have that

$$\begin{array}{ccc}
 y \triangleleft q_{(\beta)} & \xrightarrow{1_{q_{(\beta)}}} & q_{(\beta)} \\
 \downarrow \eta_q \triangleleft \epsilon_{(\beta)} & & \downarrow \epsilon_{(\beta)} \\
 q \triangleleft q & \xrightarrow{\mu_q} & q
 \end{array}$$

commutes by the unit law of the monoid (q, η_q, μ_q) .

- Suppose that α is a successor ordinal, say $\alpha = \alpha' + 1$. We must show that the following diagram commutes.

$$\begin{array}{ccc}
 y \triangleleft q_{(\beta)} + q \triangleleft q_{(\alpha')} \triangleleft q_{(\beta)} & \xrightarrow{(\iota_{\beta}, \lambda_{(\alpha'+\beta)} \circ (q \triangleleft \mu_{(\alpha', \beta)}))} & q_{(\alpha' + \beta + 1)} \\
 \downarrow (\eta_q \triangleleft \epsilon_{(\beta)}, (\mu_q \circ (q \triangleleft \epsilon_{(\alpha')}))) \triangleleft \epsilon_{(\beta)} & & \downarrow \epsilon_{(\alpha' + \beta + 1)} \\
 q \triangleleft q & \xrightarrow{\mu_q} & q
 \end{array}$$

We will show that the diagram commutes on each component of the coproduct independently. For

the first term observe that

$$\begin{array}{ccccc}
 y \triangleleft q_{(\beta)} & \xrightarrow{1_{q_{(\beta)}}} & q_{(\beta)} & \xrightarrow{l_{\beta}} & q_{(\alpha'+\beta+1)} \\
 \eta \triangleleft \epsilon_{(\beta)} \downarrow & & \downarrow \epsilon_{(\beta)} & \swarrow \epsilon_{(\alpha'+\beta+1)} & \\
 q \triangleleft q & \xrightarrow{\mu_q} & q & &
 \end{array}$$

commutes because of the unit law of the monoid (q, η_q, μ_q) and because the maps $\epsilon_{(\beta)}$ form a cocone.

For the second term consider the following diagram.

$$\begin{array}{ccccc}
 q \triangleleft q_{(\alpha')} \triangleleft q_{(\beta)} & \xrightarrow{q \triangleleft \mu_{(\alpha', \beta)}} & q \triangleleft q_{(\alpha'+\beta)} & \xrightarrow{\lambda_{(\alpha'+\beta)}} & q_{(\alpha'+\beta+1)} \\
 \downarrow q \triangleleft \epsilon_{(\alpha')} \triangleleft \epsilon_{(\beta)} & & \downarrow q \triangleleft \epsilon_{(\alpha'+\beta)} & & \downarrow \epsilon_{(\alpha'+\beta+1)} \\
 q \triangleleft q \triangleleft q & \xrightarrow{q \triangleleft \mu_q} & q \triangleleft q & & \\
 \downarrow \mu_q \triangleleft q & & \searrow \mu_q & & \\
 q \triangleleft q & \xrightarrow{\mu_q} & q & &
 \end{array}$$

The upper left square commutes by the induction hypothesis. The bottom square commutes by associativity of μ_q , and the right-most square commutes by definition of $\epsilon_{(\alpha'+\beta+1)}$. Therefore the outer square commutes as desired.

- Suppose that α is a limit ordinal. Recall that

$$q_{(\alpha)} \triangleleft q_{(\beta)} \cong \operatorname{colim}_{\alpha' < \alpha} q_{(\alpha'+\beta)}.$$

The composite $\epsilon_{(\alpha+\beta)} \circ \mu_{(\alpha, \beta)}$ is induced by the cocone defined for $\alpha' < \alpha$ by

$$q_{(\alpha')} \triangleleft q_{(\beta)} \xrightarrow{\mu_{(\alpha', \beta)}} q_{(\alpha'+\beta)} \xrightarrow{l_{(\alpha'+\beta)}} q_{(\alpha+\beta)} \xrightarrow{\epsilon_{(\alpha+\beta)}} q$$

while the composite $\mu_q \circ (\epsilon_{(\alpha)} \triangleleft \epsilon_{(\beta)})$ is induced by the cocone defined for $\alpha' < \alpha$ by

$$q_{(\alpha')} \triangleleft q_{(\beta)} \xrightarrow{\epsilon_{(\alpha')} \triangleleft \epsilon_{(\beta)}} q \triangleleft q \xrightarrow{\mu_q} q.$$

For each $\alpha' < \alpha$ these composites are identical by the induction hypothesis. Therefore by uniqueness of the induced maps

$$\epsilon_{(\alpha+\beta)} \circ \mu_{(\alpha, \beta)} = \mu_q \circ (\epsilon_{(\alpha)} \triangleleft \epsilon_{(\beta)})$$

as desired. □

Theorem 2.11. *There is an adjunction*

$$\mathbf{Poly} \begin{array}{c} \xrightarrow{m_-} \\ \Rightarrow \\ \xleftarrow{U} \end{array} \mathbf{Mod(Poly)} .$$

Proof. For $p : \mathbf{Poly}$ define the unit of the adjunction, $\zeta_p : p \rightarrow \mathfrak{m}_p$, to be the composite

$$p \xrightarrow{\lambda_{(0)}} p_{(1)} \xrightarrow{\iota_{(1)}} \mathfrak{m}_p.$$

For $f : p \rightarrow q$ in $\mathbf{Poly}$, consider the following diagram.

$$\begin{array}{ccccc} p & \xrightarrow{\lambda_{(0)}} & y + p = p_{(1)} & \xrightarrow{\iota_{(1)}} & \mathfrak{m}_p \\ f \downarrow & & \downarrow y+f=f_{(1)} & & \downarrow \mathfrak{m}_f \\ q & \xrightarrow{\lambda_{(0)}} & y + q = q_{(1)} & \xrightarrow{\iota_{(1)}} & \mathfrak{m}_q \end{array}$$

The square on the left commutes by definition of $f_{(1)}$ while the square on the right commutes by the commuting diagram in Equation (5). Therefore, the outer square commutes and ζ is natural.

For the $\triangleleft$ -monoid (q, η_q, μ_q) , we define the counit of the adjunction $\epsilon_q : (\mathfrak{m}_q, \eta, \mu) \rightarrow (q, \eta_q, \mu_q)$ to be $\epsilon_{(\kappa)} : \mathfrak{m}_q \rightarrow q$ for κ such that q is κ -small. The map ϵ_q preserves the unit because the maps $\epsilon_{(\alpha)}$ form a cocone. That ϵ_q preserves multiplication is a direct result of Lemma B.2. Therefore ϵ_q is indeed a map of $\triangleleft$ -monoids.

Next we want to show that ϵ is natural. Let $f : (p, \eta_p, \mu_p) \rightarrow (q, \eta_q, \mu_q)$ in $\mathbf{Mod}(\mathbf{Poly})$. It suffices to show for all α the following diagram commutes.

$$\begin{array}{ccc} p_{(\alpha)} & \xrightarrow{f_{(\alpha)}} & q_{(\alpha)} \\ \epsilon_{(\alpha)} \downarrow & & \downarrow \epsilon_{(\alpha)} \\ p & \xrightarrow{f} & q \end{array}$$

We show this by induction on α . For $\alpha = 0$, $f_{(\alpha)}$ is the identity on y and the diagram commutes because f preserves the unit.

Suppose that the diagram commutes for all $\alpha' < \alpha$. If α is a successor ordinal — say $\alpha = \alpha' + 1$ — then we want to show that the following diagram commutes.

$$\begin{array}{ccc} y + p \triangleleft p_{(\alpha')} & \xrightarrow{y+f \triangleleft f_{(\alpha')}} & y + q \triangleleft q_{(\alpha')} \\ (\eta_p, \mu_p \circ (p \triangleleft \epsilon_{(\alpha')})) \downarrow & & \downarrow (\eta_q, \mu_q \circ (q \triangleleft \epsilon_{(\alpha')})) \\ p & \xrightarrow{f} & q \end{array}$$

It commutes on the first term of the coproduct, again because f preserves the unit. To show that it commutes on the second term, we want to show that the outer diagram in the following commutes.

$$\begin{array}{ccc} p \triangleleft p_{(\alpha')} & \xrightarrow{f \triangleleft f_{(\alpha')}} & q \triangleleft q_{(\alpha')} \\ p \triangleleft \epsilon_{(\alpha')} \downarrow & & \downarrow q \triangleleft \epsilon_{(\alpha')} \\ p \triangleleft p & \xrightarrow{f \triangleleft f} & q \triangleleft q \\ \mu_p \downarrow & & \downarrow \mu_q \\ p & \xrightarrow{f} & q \end{array}$$

The top square commutes by the induction hypothesis and the bottom square commutes because f preserves multiplication.

If α is a limit ordinal, consider the cocones

$$p_{(\alpha')} \xrightarrow{f_{(\alpha')}} q_{(\alpha')} \xrightarrow{\epsilon_{(\alpha')}} q \quad \text{and} \quad p_{(\alpha')} \xrightarrow{\epsilon_{(\alpha')}} p \xrightarrow{f} q.$$

By uniqueness, the cocone on the left induces the map $p_{(\alpha)} \xrightarrow{f_{(\alpha)}} q_{(\alpha)} \xrightarrow{\epsilon_{(\alpha)}} q$. Also by uniqueness, the cocone on the right induces the map $p_{(\alpha)} \xrightarrow{\epsilon_{(\alpha)}} p \xrightarrow{f} q$. Furthermore, these cocones are identical by the induction hypothesis and so their induced maps are equal, as desired.

Finally, to show that the unit and counit form an adjunction we must show that for a polynomial p

$$\mathfrak{m}_p \xrightarrow{\mathfrak{m}_{\zeta_p}} \mathfrak{m}_{\mathfrak{m}_p} \xrightarrow{\epsilon_{\mathfrak{m}_p}} \mathfrak{m}_p$$

is the identity.

It suffices to show that for all α ,

$$p_{(\alpha)} \xrightarrow{(\zeta_p)_{(\alpha)}} (\mathfrak{m}_p)_{(\alpha)} \xrightarrow{\epsilon_{(\alpha)}} \mathfrak{m}_p$$

is the inclusion $\iota_{(\alpha)}$. We do this by transfinite induction. The base case and the induction step for limit ordinals follow directly from definitions. For successor ordinals $\alpha + 1$, it suffices to show that the following diagram commutes:

$$\begin{array}{ccccc} y + p \triangleleft p_{(\alpha)} & \xrightarrow{y + p \triangleleft \iota_{(\alpha)}} & y + p \triangleleft \mathfrak{m}_p & & \\ \parallel & & & & \parallel \\ y + p \triangleleft p_{(\alpha)} & \xrightarrow{y + p \triangleleft (\zeta_p)_{(\alpha)}} & y + p \triangleleft (\mathfrak{m}_p)_{(\alpha)} & \xrightarrow{y + p \triangleleft \epsilon_{(\alpha)}} & y + p \triangleleft \mathfrak{m}_p \\ \parallel & & \downarrow y + \zeta_p \triangleleft (\mathfrak{m}_p)_{(\alpha)} & & \uparrow \cong \\ p_{(\alpha+1)} & \xrightarrow{(\zeta_p)_{(\alpha+1)}} & y + \mathfrak{m}_p \triangleleft (\mathfrak{m}_p)_{(\alpha)} & \xrightarrow{\epsilon_{(\alpha+1)}} & \mathfrak{m}_p \end{array}$$

The top square commutes by the induction hypothesis. The bottom left and right squares commute by definitions of $(\zeta_p)_{(\alpha+1)}$ and $\epsilon_{(\alpha+1)}$, respectively.

We must also show that for a $\triangleleft$ -monoid (q, η_q, μ_q)

$$q \xrightarrow{\zeta_q} \mathfrak{m}_q \xrightarrow{\epsilon_q} q$$

is the identity as well.

Since the diagrams below commutes, it suffices to show that $\epsilon_{(1)} \circ \lambda_{(0)}$ is the identity. This is immediate from the definition of $\epsilon_{(1)}$ and the unit law for (q, η_q, μ_q) .

$$\begin{array}{ccccc} q & \xrightarrow{\lambda_{(0)}} & q_{(1)} & \xrightarrow{\iota_{(1)}} & \mathfrak{m}_q \xrightarrow{\epsilon_q} q \\ & & & \searrow \epsilon_{(1)} & \nearrow \end{array}$$

□

C Proofs for the Cofree Comonad Monad

Proposition 3.1. *There is a functor $\mathsf{c}_- : \mathbf{Poly} \rightarrow \mathbf{Poly}$ such that c_p has the structure of a $\triangleleft$ -comonoid for each $p : \mathbf{Poly}$,*

$$\mathsf{c}_p \rightarrow y \quad \text{and} \quad \mathsf{c}_p \rightarrow \mathsf{c}_p \triangleleft \mathsf{c}_p.$$

Proof. Given a polynomial p , define polynomials $p^{(i)}$ for $i \in \mathbb{N}$ by

$$p^{(0)} := y \quad \text{and} \quad p^{(1+i)} := y \times (p \triangleleft p^{(i)})$$

There is a projection map $\pi^{(0)} : p^{(1)} \rightarrow p^{(0)}$, and if $\pi^{(i)} : p^{(1+i)} \rightarrow p^{(i)}$ has been defined, then we can define $\pi^{(1+i)} := y \times (p \triangleleft \pi^{(i)})$. Now define the polynomial

$$\mathsf{c}_p := \lim \left(\cdots \xrightarrow{\pi^{(2)}} p^{(2)} \xrightarrow{\pi^{(1)}} p^{(1)} \xrightarrow{\pi^{(0)}} p^{(0)} \right) \quad (10)$$

and we note that this construction $p \mapsto \mathsf{c}_p$ is natural in $p : \mathbf{Poly}$.

This polynomial comes equipped with a counit $\epsilon : \mathsf{c}_p \rightarrow y = p^{(0)}$ given by the projection. We next construct the comultiplication $\delta : \mathsf{c}_p \rightarrow \mathsf{c}_p \triangleleft \mathsf{c}_p$. Since $\triangleleft$ commutes with connected limits, we have

$$\mathsf{c}_p \triangleleft \mathsf{c}_p = \left(\lim_{i_1} p^{(i_1)} \right) \triangleleft \left(\lim_{i_2} p^{(i_2)} \right) \cong \lim_{i_1, i_2} \left(p^{(i_1)} \triangleleft p^{(i_2)} \right)$$

To obtain the comultiplication $\lim_i p^{(i)} \rightarrow \lim_{i_1, i_2} (p^{(i_1)} \triangleleft p^{(i_2)})$, it suffices to produce a natural choice of polynomial map $\varphi_{i_1, i_2} : p^{(i_1+i_2)} \rightarrow p^{(i_1)} \triangleleft p^{(i_2)}$ for any $i_1, i_2 : \mathbb{N}$. When $i_1 = 0$ or $i_2 = 0$, we use the unit identity for $\triangleleft$. By induction, assume given $\varphi_{i_1, 1+i_2}$; we construct $\varphi_{1+i_1, 1+i_2}$ as follows:

$$\begin{aligned} p^{(1+i_1+1+i_2)} &= y \times (p \triangleleft p^{(i_1+1+i_2)}) \\ &\rightarrow y \times (p \triangleleft p^{(i_1)} \triangleleft p^{(1+i_2)}) \end{aligned} \quad (11)$$

$$\begin{aligned} &\rightarrow (y \times p \triangleleft p^{(i_1)}) \triangleleft p^{(1+i_2)} \\ &= p^{(1+i_1)} \triangleleft p^{(1+i_2)} \end{aligned} \quad (12)$$

where (11) is $\varphi_{i_1, 1+i_2}$ and it remains to construct (12). Since $- \triangleleft q$ preserves products for any q , constructing (12) is equivalent to constructing two maps

$$y \times (p \triangleleft p^{(i_1)} \triangleleft p^{(1+i_2)}) \xrightarrow{\phi^{(i_1, i_2)}} p^{(1+i_2)} \quad \text{and} \quad y \times (p \triangleleft p^{(i_1)} \triangleleft p^{(1+i_2)}) \rightarrow p \triangleleft p^{(i_1)} \triangleleft p^{(1+i_2)}.$$

For the latter we use the second projection. The former, $\phi^{(i_1, i_2)} : p^{(1+i_1+1+i_2)} \rightarrow p^{(1+i_2)}$, is the more interesting one; for it we also use projections $p^{(i_1)} \rightarrow p^{(0)} = y$ and $\pi^{(i_2)} : p^{(i_2+1)} \rightarrow p^{(i_2)}$ to obtain:

$$y \times (p \triangleleft p^{(i_1)} \triangleleft p^{(1+i_2)}) \rightarrow y \times (p \triangleleft y \triangleleft p^{(i_2)}) \cong p^{(1+i_2)}$$

We leave the naturality of this to the reader.

It remains to check that ϵ and δ satisfy unitality and coassociativity. The base cases above imply unitality. Proving coassociativity amounts to proving that the following diagram commutes:

$$\begin{array}{ccc} p^{(1+i_1+1+i_2+1+i_3)} & \xrightarrow{\phi^{(i_1, i_2+1+i_3)}} & p^{(1+i_2+1+i_3)} \\ \phi^{(i_1+1+i_2, i_3)} \downarrow & & \downarrow \phi^{(i_2, i_3)} \\ p^{(1+i_3)} & \xlongequal{\quad\quad\quad} & p^{(1+i_3)} \end{array}$$

This can be shown by induction on i_3 . □

Proposition C.1. *There is a monoidal structure on $c : \mathbf{Poly} \rightarrow \mathbf{Poly}$*

$$y \rightarrow c_y \quad \text{and} \quad c_p \otimes c_q \rightarrow c_{p \otimes q}.$$

Proof. The polynomial $c_y \cong y^{\mathbb{N}}$ has a unique position, and this defines the first map. However, it is conceptually cleaner to realize that comonads are closed under $\otimes$ by duoidality, and hence both y and $c_p \otimes c_q$ carry comonad structures. Thus the desired maps are induced by the obvious polynomial maps $y \cong y$ and $c_p \otimes c_q \rightarrow p \otimes q$. It is straightforward to check that these are unital and associative. □

Theorem 3.2. *There is an adjunction*

$$\mathbf{Cat}^\# \begin{array}{c} \xrightarrow{U} \\ \xRightarrow{\quad} \\ \xleftarrow{c_-} \end{array} \mathbf{Poly}.$$

Proof. We will abuse notation and denote the comonoid $(c, \epsilon, \delta) : \mathbf{Cat}^\#$ simply by its carrier c . We first provide the counit and unit of the desired adjunction. The counit

$$\epsilon_p : c_p \rightarrow p$$

is given by composing the projection map $c_p \rightarrow p^{(1)}$ from construction (10) with the projection $p^{(1)} \cong y \times p \rightarrow p$. Since c_c is defined as a limit, the unit

$$\eta_c : c \rightarrow c_c$$

will be given by defining maps $\eta^{(i)} : c \rightarrow c^{(i)}$ commuting with the projections $\pi^{(i)} : c^{(1+i)} \rightarrow c^{(i)}$, for each $i : \mathbb{N}$, and then showing that the resulting polynomial map η_c is indeed a cofunctor. Noting that $c^{(0)} = y$, we define

$$\eta^{(0)} := \epsilon$$

Given $\eta^{(i)} : c \rightarrow c^{(i)}$, we define $\eta^{(1+i)}$ as the composite

$$c \xrightarrow{(\epsilon, \delta)} y \times (c \triangleleft c) \xrightarrow{y \times (c \triangleleft \eta^{(i)})} y \times (c \triangleleft c^{(i)}) = c^{(1+i)}.$$

Clearly, we have $\eta^{(0)} = \pi^{(0)} \circ \eta^{(1)}$. It is easy to check that if $\eta^{(i)} = \pi^{(i)} \circ \eta^{(1+i)}$ then $\eta^{(1+i)} = \pi^{(1+i)} \circ \eta^{(2+i)}$. Thus we have constructed a polynomial map $\eta : c \rightarrow c_c$. It clearly commutes with the counit, so it suffices

to show that η commutes with the comultiplication, which amounts to showing that the following diagram commutes

$$\begin{array}{ccccc}
 c & \xrightarrow{\delta} & c \triangleleft c & \xrightarrow{(\epsilon, \delta) \triangleleft (\epsilon, \delta)} & (y \times (c \triangleleft c)) \triangleleft (y \times (c \triangleleft c)) \\
 (\epsilon, \delta) \downarrow & & & & \downarrow (y \times c \triangleleft \eta^{(i_1)}) \triangleleft (y \times c \triangleleft \eta^{(i_2)}) \\
 y \times (c \triangleleft c) & \xrightarrow[y \times c \triangleleft \eta^{(i_1+1+i_2)}]{} & y \times (c \triangleleft c^{(i_1+1+i_2)}) & \xrightarrow[\varphi_{1+i_1, 1+i_2}]{} & (y \times (c \triangleleft c^{(i_1)})) \triangleleft (y \times (c \triangleleft c^{(i_2)}))
 \end{array}$$

for all $i_1, i_2 : \mathbb{N}$, where $\varphi_{1+i_1, 1+i_2}$ is the map constructed in Equation (11) and Equation (12). Commutativity follows from the counitality and coassociativity of the comonoid c .

The triangle identities are straightforward as well. Indeed, for any comonoid $c : \mathbf{Cat}^\sharp$, the composite $c \xrightarrow{U \circ \eta_c} c_c \xrightarrow{\epsilon_{Uc}} c$ is equal to the composite of $c \xrightarrow{(\epsilon, c)} c^{(1)} = y \times c$, with the projection $c^{(1)} \rightarrow c$, the result of which is the identity. Finally, for any polynomial $p : \mathbf{Poly}$, the composite $c_p \xrightarrow{\eta_{c_p}} c_{c_p} \xrightarrow{\epsilon_{c_p}} c_p$ is given by taking a limit of maps of the form

$$c_p \xrightarrow{(\epsilon, \delta)} y \times (c_p \triangleleft c_p) \xrightarrow{y \times (c_p \triangleleft \eta^{(i)})} y \times (c_p \triangleleft c_p^{(i)}) \xrightarrow{y \times (\epsilon_p \triangleleft \epsilon_p^{(i)})} y \times (p \triangleleft p^{(i)})$$

Each one is in fact the projection $c_p \rightarrow p^{(i+1)}$, so the resulting map is the identity on c_p , completing the proof. $\square$

D Proofs for the Module Structure $m_p \otimes c_q \rightarrow m_{p \otimes q}$

Proposition 3.3. *The maps $\Xi_{p,q}$ are natural in p and q .*

Proof. To show that ψ is natural in p , it suffices to show that for a maps $p \rightarrow p'$ and $q \rightarrow q'$ in $\mathbf{Poly}$ the following diagram commutes:

$$\begin{array}{ccc}
 p & \longrightarrow & [c_q, m_{p \otimes q}] \\
 \downarrow & & \downarrow \\
 p' & \longrightarrow & [c_{q'}, m_{p' \otimes q'}]
 \end{array}$$

This follows immediately from the commutativity of the following diagram.

$$\begin{array}{ccccc}
 p \otimes c_q & \longrightarrow & p \otimes q & \longrightarrow & m_{p \otimes q} \\
 \downarrow & & \downarrow & & \downarrow \\
 p' \otimes c_{q'} & \longrightarrow & p' \otimes q' & \longrightarrow & m_{p' \otimes q'}
 \end{array}$$

Note that the square on the left commutes by naturality of the counit of the adjunction in Theorem 3.2. The square on the right commutes by naturality of unit of the adjunction in Theorem 2.11. $\square$

Theorem 3.4. *There is a left-module over $c_- : (\mathbf{Poly}, \otimes, y) \rightarrow (\mathbf{Poly}, \otimes, y)$ consisting of:*

- $\mathbf{Poly}$ as a left module category over $(\mathbf{Poly}, \otimes, y)$.
- The functor $m_- : \mathbf{Poly} \rightarrow \mathbf{Poly}$.
- The natural transformation $\Xi : m_- \otimes c_- \Rightarrow m_{-\otimes-}$.

Proof. We must show that two diagrams commute. First, we will show that the following diagram commutes.

$$\begin{array}{ccc} y \otimes m_r & \xrightarrow{\cong} & m_r \\ \downarrow & & \uparrow \cong \\ c_y \otimes m_r & \xrightarrow{\Xi_{y,r}} & m_{y \otimes r} \end{array}$$

It suffices to show that the diagram below commutes:

$$\begin{array}{ccccc} y \otimes r & \longrightarrow & r & \longrightarrow & m_r \\ \downarrow & & \uparrow & & \uparrow \\ c_y \otimes r & \longrightarrow & y \otimes r & \longrightarrow & m_{y \otimes r} \end{array}$$

The square on the left commutes because $y \rightarrow c_y \rightarrow y$ is the identity. The square on the right commutes by the naturality of the unit of the adjunction in Theorem 2.11.

Next we must show that the following diagram commutes.

$$\begin{array}{ccc} c_p \otimes c_q \otimes m_r & \longrightarrow & c_p \otimes m_{q \otimes r} \\ \downarrow & & \downarrow \\ c_{p \otimes q} \otimes m_r & \longrightarrow & m_{p \otimes q \otimes r} \end{array}$$

It suffices to show that the following diagram commutes.

$$\begin{array}{ccccc} c_p \otimes c_q \otimes r & \longrightarrow & c_p \otimes q \otimes r & \longrightarrow & c_p \otimes m_{q \otimes r} \\ \downarrow & & \downarrow & & \downarrow \Xi_{p,q \otimes r} \\ c_{p \otimes q} \otimes r & \longrightarrow & p \otimes q \otimes r & \longrightarrow & m_{p \otimes q \otimes r} \end{array}$$

The square on the left commutes by definition of the laxator of c_- while the square on the right commutes by definition of $\Xi_{p,q \otimes r}$. $\square$

Abstract Kleisli Structures on 2-categories*

Adrian Miranda

Department of Mathematics
University of Manchester
Manchester
United Kingdom

adrian.miranda@manchester.ac.uk

Führmann introduced Abstract Kleisli structures to model call-by-value programming languages with side effects, and showed that they correspond to monads satisfying a certain equalising condition on the unit. We first extend this theory to non-strict morphisms of monads, and to incorporate 2-cells of monads. We then further extend this to a theory of abstract Kleisli structures on 2-categories, characterising when the original pseudomonad can be recovered by the abstract Kleisli structure on its 2-category of free-pseudoalgebras.

1 Introduction

1.1 Context and motivation

Abstract Kleisli structures, also known as thunk-force categories, axiomatise structure that one finds on the Kleisli category of any monad, and have been used to provide direct models of the computational λ -calculus [5]. Their duals, which axiomatise structure on the coKleisli category of any comonad, have also found applications to runnable monads [3] and the theory of cartesian differential categories [13]. Variants such as cartesian reverse differential categories build upon the latter of these and are used in modern categorical treatments of gradient-based learning [2]. Mathematically, abstract Kleisli structures capture precisely those monads whose unit $\eta : 1_B \Rightarrow T$ is the equaliser of $T\eta$ and η_T . This condition is also equivalent to saying that the Eilenberg Moore adjunction of the monad is of *codescent type* which means that the comparison from B to the category of coalgebras for the comonad induced on B^T is fully-faithful.

Pseudomonads generalise monads to the two-dimensional setting by allowing conditions such as naturality and monad laws to hold up to isomorphism. They have also received attention in computer science [17], where their 2-cells allow outputs of computations to be considered before being rewritten or identified in a normal form. However, as yet abstract Kleisli structures remain unexplored in the two-dimensional context. We fill this gap in the literature, and lay the mathematical foundations for future work on the two-dimensional λ -calculus and differential λ -calculus [4].

1.2 Outline

Section 2 reviews and extends the main results on abstract Kleisli structures in the one-dimensional setting (Theorem 5.3, Lemmas 5.6, 5.28 and 5.29 of [5]). These results exhibit abstract Kleisli structures

*This material is based on Chapter 8 of my PhD thesis [16]. I am grateful for having been supported by the MQRES PhD Scholarship 20192497 while conducting this research, and by EPSRC under grant EP/V002325/2 while writing the paper. I thank Richard Garner for suggesting this as a research topic, Steve Lack for his guidance while I was conducting this research, Nicola Gambino and Mike Johnson for their guidance while I was preparing this paper, and J. S. Lemay for his encouragement.

as objects of a reflective full-subcategory of a category of monads and characterise the monads which correspond to abstract Kleisli structures. We contribute a new, concise definition of abstract Kleisli structures and their categories of ‘thinkable morphisms’, using a lifting condition. We also extend the results of [5] in two directions. The first direction of generalisation is from strict morphisms of monads to the more general morphisms of monads introduced in [19] which are in one-to-one correspondence with extensions between Kleisli categories. The second is to describe the two kinds of 2-cells between abstract Kleisli structures, namely the monad transformations and the more general 2-cells considered in [11].

The remaining sections extend this theory to the 2-categorical setting. Section 3 defines abstract Kleisli structures on 2-categories, and thinkability in the two-dimensional setting. Section 4 describes a bicategorical limit condition on the unit which is shown in Theorem 6.1 to characterise those pseudomonads that are recoverable from the abstract Kleisli structures on their 2-categories of free pseudoalgebras. We prove some intermediate results towards this goal in Section 5. Finally, in Theorem 6.8 we exhibit abstract Kleisli structures on 2-categories as a full, reflective sub-**Gray** category of two suitable **Gray**-categories of pseudomonads.

1.3 Assumed knowledge, conventions and techniques

We assume familiarity with pseudomonads and their 2-categories of pseudoalgebras, fixing notation in Definition 3.1 and referring to [15] for details. Many of our proofs require chasing large pasting diagrams. These are typically omitted for the sake of brevity but can be found in either Chapter 8 or the appendices of the author’s Ph. D. thesis [16]. Some sample calculations are included in Appendix 8 for illustrative purposes. In these proofs we freely use the pasting theorem for 2-categories [18]. A particular bicategorical limit [9] called an isobidescent object will be needed, and its relevant properties will be described explicitly in Section 4. The results of Section 6 are expressed in terms of the semi-strict three-dimensional categories [8], or **Gray**-categories, that pseudomonads form. One of these structures is the Kleisli version of the **Gray**-category of pseudomonads defined in [14], while the other extends this structure in the spirit of [11]. Here **Gray** denotes **2-Cat** equipped with the **Gray**-tensor product [7], and a **Gray**-category is a category enriched over this base.

2 Abstract Kleisli structures on categories

We begin with a reformulation of the definition of abstract Kleisli structures in Definition 2.1 and the corresponding monad on the category of thinkable morphisms in Proposition 2.4. We then recall 2-categories of monads from [11] in Notation 2.5, and use this to define morphisms and 2-cells of abstract Kleisli structures in Definition 2.6. In contrast, no 2-cells of abstract Kleisli structures are defined in [5] while their morphisms of abstract Kleisli structures commute with all structure on the nose and correspond to strict morphisms of monads.

Definition 2.1.

1. An *abstract Kleisli structure* on a category B consists of
 - A comonad (Q, ε, δ) on B .
 - A functor $\theta : B_0 \rightarrow B^Q$ providing a lifting as in the following diagram, where the unlabelled horizontally depicted functors include the discrete category on the set of objects of B , and $U^Q \dashv F^Q$ is the co-Eilenberg-Moore adjunction for $(B, Q, \varepsilon, \delta)$.

$$\begin{array}{ccccc}
B_0 & \xrightarrow{\quad} & B & \xrightarrow{F^Q} & B^Q \\
\downarrow Q_0 & & & \nearrow \theta & \downarrow U^Q \\
& & & & B \\
& & & & \downarrow \\
& & & & B
\end{array}$$

2. Given an abstract Kleisli structure on B , the associated *category of thunkable morphisms* is given as the factorisation of θ as displayed below, in which K is fully faithful and θ' is bijective on objects.

$$B_0 \xrightarrow{\theta'} B_\theta \xrightarrow{K} B^Q$$

Example 2.2. Let (A, S, η, μ) be a monad. Then the Kleisli category A_S inherits an abstract Kleisli structure with (Q, ε, δ) the comonad induced by the Kleisli adjunction and $\theta_X := F_S \eta_X$. This captures all examples, and gives the concept its name.

Remark 2.3. Definition 2.1 part (1) indeed recaptures Definition 2.1 of [5]. The latter consists of a co-pointed endofunctor (B, Q, ε) , an unnatural transformation $\theta : 1_B \rightrightarrows Q$ and various axioms amounting to comonad laws for $(Q, \varepsilon, \theta_Q)$ and coalgebra laws for $\theta_X : X \rightarrow QX$. The commutativity of the top triangle in Definition 2.1 part (1) amounts to $\delta = \theta_Q$, while the morphisms in the intermediate category B_θ constructed in Definition 2.1 part (2) are indeed the thunkable morphisms described in Definition 2.3 of [5]; the condition for $f : (X, \theta_X) \rightarrow (Y, \theta_Y)$ to be a morphism of coalgebras is precisely naturality of the assignment $X \mapsto \theta_X$ in the morphism $f : X \rightarrow Y$.

Proposition 2.4. The composite functor $F_\theta := B_\theta \xrightarrow{K} B^Q \xrightarrow{U^Q} B$ is faithful and has a right adjoint U_θ , such that the comonad induced on B is (Q, ε, δ) .

Proof. First observe that forgetful functors from categories of coalgebras are faithful, and K is faithful by construction, so the composite F_θ is also faithful. The right adjoint acts as $U_\theta(f : X \rightarrow Y) = Qf : QX \rightarrow QY$, with these outputs being morphisms of free coalgebras and hence in B_θ . The unit of the adjunction is given by θ , which is itself in B_θ by the coassociativity axiom for each coalgebra (X, θ_X) . Naturality for θ as a unit for the adjunction holds by construction of B_θ , while right triangle identity holds in B_θ by the right unit law for $(Q, \varepsilon, \theta_Q)$ and the left triangle identity holds in B by the unit law for (X, θ_X) as a coalgebra. Finally, since $\delta = \theta_Q$, we see that the comonad induced on B is indeed (Q, ε, δ) . $\square$

Notation 2.5. For $\kappa \in \{\tau, \lambda\}$, the 2-category $\mathbf{Monads}_\kappa$ has objects given by monads and morphisms $(A, S) \rightarrow (B, T)$ given by pairs of functors $F : A \rightarrow B$ and $\overline{F} : A_S \rightarrow B_T$ commuting with Kleisli left adjoints. These will be referred to as co-morphisms of monads. A 2-cell $(\phi, \overline{\phi}) : (F, \overline{F}) \Rightarrow (G, \overline{G})$ in $\mathbf{Monads}_\tau$ consists of a pair of natural transformations $\phi : F \Rightarrow G$ and $\overline{\phi} : \overline{F} \Rightarrow \overline{G}$ satisfying a commutativity condition with the left adjoints. Meanwhile, a 2-cell in $\mathbf{Monads}_\lambda$ just consists of natural transformation between the Kleisli categories. The 2-functor $\mathbf{Monads} : \mathbf{Monads}_\tau \rightarrow \mathbf{Monads}_\lambda$ is similar to the one described in 2.1 of [11] with Kleisli categories instead of Eilenberg-Moore categories. If a 2-category is denoted with the subscript τ (resp. λ) then its 2-cells will be called tight (resp. loose).

Definition 2.6. Let $\mathbf{AbsKL}_0$ be the class of abstract Kleisli structures and let $\tau : \mathbf{AbsKL}_0 \rightarrow \mathbf{Monads}_\tau$ be the class function which sends an abstract Kleisli structure to the monad on its category of thunkable morphisms as per Proposition 2.4. The 2-category $\mathbf{AbsKL}_\tau$ is defined as the image of τ , while the 2-category $\mathbf{AbsKL}_\lambda$ is defined as the image of the composite $\mathbf{AbsKL}_0 \xrightarrow{\tau} \mathbf{Monads}_\tau \xrightarrow{\mathbf{Monads}} \mathbf{Monads}_\lambda$.

This perspective is used to define **Gray**-categories of abstract Kleisli structures on 2-categories in Definitions 6.2 and 6.3. Proposition 2.7, to follow, re-expresses the data of Definition 2.6 in terms of compatibility with the data in an abstract Kleisli structure.

Proposition 2.7. *Let (A, P, π) and (B, Q, θ) be abstract Kleisli structures and let $F_\pi : A_\pi \rightarrow A$ and $F_\theta : B_\theta \rightarrow B$ be the left adjoints described in Proposition 2.4. Let $\bar{G} : A \rightarrow B$ be a functor.*

1. *To give $G : A_\pi \rightarrow B_\theta$ such that $(G, \bar{G})$ is a morphism of abstract Kleisli structures is to assert that $\bar{G}$ preserves thunkability. That is, if $f : X \rightarrow Y$ satisfies π -naturality then $\bar{G}f$ satisfies θ -naturality.*
2. *Given $(H, \bar{H}) : (A, P, \pi) \rightarrow (B, Q, \theta)$ another morphism of abstract Kleisli structures, to give a loose 2-cell $\phi : (G, \bar{G}) \Rightarrow (H, \bar{H})$ is just to give a natural transformation $\bar{\phi} : \bar{G} \Rightarrow \bar{H}$.*
3. *Given $\bar{\phi}$ as in part (2), to give a ϕ making $(\phi, \bar{\phi})$ into a tight 2-cell is to assert that the components $\bar{\phi}_X$ are thunkable.*
4. *There is a commutative square of 2-functors as depicted below, in which the horizontal maps are 2-fully faithful and send an abstract Kleisli structure (B, Q, θ) to the monad induced on B_θ from the adjunction described in Proposition 2.4.*

$$\begin{array}{ccc}
 \mathbf{AbsKL}_\tau & \xrightarrow{\quad} & \mathbf{Monads}_\tau \\
 \mathbf{AbsKL} \downarrow & & \downarrow \mathbf{Monads} \\
 \mathbf{AbsKL}_\lambda & \xrightarrow{\quad} & \mathbf{Monads}_\lambda
 \end{array}$$

Proof. Parts (1), (2) and (3) are easy to observe using bijectivity on objects and faithfulness of the left adjoints. Part (4) follows by construction. $\square$

Remark 2.8. Although it will not be needed for any of our proofs, we note that Proposition 2.7 part (4) is a fully faithful **BO**-enriched functor, in the sense of [10].

Theorem 2.9. *Let (B, T, η, μ) be a monad, $\underline{T}$ be the comonad induced on the Kleisli category B_T , and $\bar{T}$ be the comonad induced on the Eilenberg-Moore category B^T . The following are equivalent.*

1. *(B, T, η, μ) is in the essential image of $I_\kappa : \mathbf{AbsKL}_\kappa \rightarrow \mathbf{Monads}_\kappa$ for $\kappa \in \{\tau, \lambda\}$.*
2. *The natural transformation η is the equaliser of $T\eta$ and η_T .*
3. *The Kleisli left adjoint $F_T : B \rightarrow B_T$ is both faithful and full on thunkable morphisms.*
4. *The canonical comparison $B \rightarrow (B^T)^{\bar{T}}$ is fully faithful.*
5. *The canonical comparison $B \rightarrow (B_T)^{\underline{T}}$ is fully faithful.*

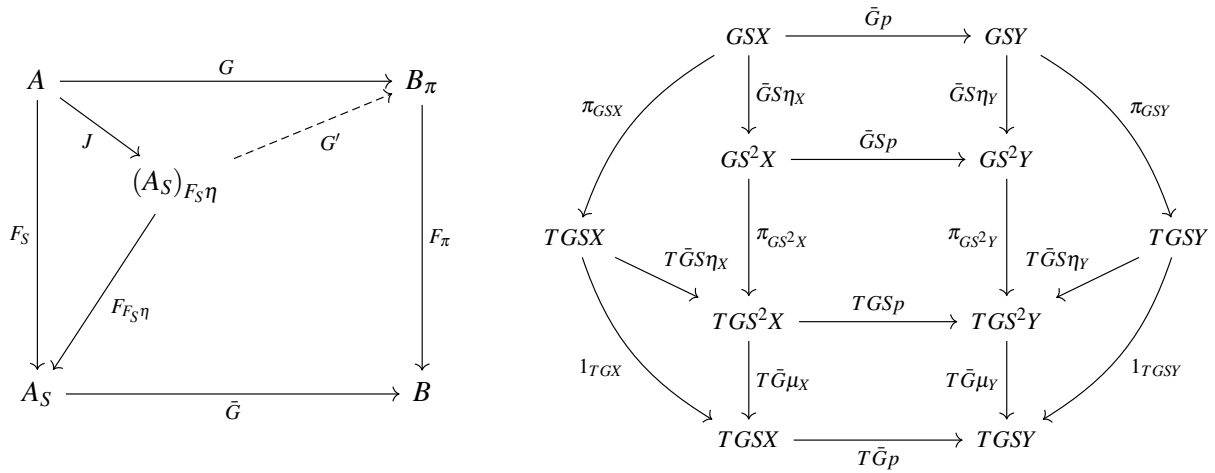
Proof. (1) $\iff$ (2) is Lemma 5.2.8 of [5]. (2) $\iff$ (3) is Lemma 5.2.7 of [5]. (4) $\iff$ (5) is clear since the image of any $X \in B$ under the canonical comparison $B \rightarrow (B^T)^{\bar{T}}$ is a coalgebra for $\bar{T}$ whose underlying algebra for T is free, and is hence also a coalgebra for $\underline{T}$. (2) $\iff$ (4) is a standard result; see Corollary 7 and Theorem 9 of [1]. $\square$

Theorem 2.10. *Let (A, S) be a monad, let $(A_S, \bar{S}, F_{S\eta})$ be the abstract Kleisli structure of Example 2.2 and let $\bar{\bar{S}}$ be the monad induced on $(A_S)_{F_{S\eta}}$. Then*

1. *$F_S : A \rightarrow A_S$ factorises through the left adjoint $F_{F_{S\eta}} : (A_S)_{F_{S\eta}} \rightarrow A_S$ via a functor J .*

2. $(J, 1_{A_S}) : (A, S) \rightarrow ((A_S)_{F_S\eta}, \bar{S})$ is a co-morphism of monads.
3. The co-morphism of monads $(J, 1_{A_S})$ has the universal property of a unit exhibiting that there are left adjoints to $I_\kappa : \mathbf{AbsKL}_\kappa \rightarrow \mathbf{Monads}_\kappa$ for $\kappa \in \{\tau, \lambda\}$.

Proof. For part (1), to give J is simply to note that $F_S(f : X \rightarrow Y)$ is always thunkable. Part (2) follows immediately from Part (1). For part (3), let (B, T, π) be an abstract Kleisli category. We first consider the one-dimensional aspect of the universal property for $(J, 1_{A_S})$ as a unit exhibiting a left adjoint to I . By faithfulness of the left adjoints and fully-faithfulness of I , it suffices to give a H' as in the diagram below left, for which in turn it suffices to show that H preserves thunkability. Preservation of thunkability can be seen by commutativity of the diagram below right. Note that $\bar{G}Sf = Gf$ for any morphism f in A , and that Gf is thunkable by assumption.



The universal property holds trivially with respect to loose 2-cells, as they are merely natural transformations between the Kleisli categories and do not need to be factorised. Finally for tight 2-cells $(\phi, \bar{\phi})$, it suffices to see that ϕ is natural with respect to thunkable morphisms in A_S . But this is true since $\bar{\phi}$ is natural with respect to all morphisms in A_S and $F_\pi \cdot \phi = \bar{\phi} \cdot F_S$. $\square$

3 Categorified thunkability

We now categorify the notion of abstract Kleisli structures and their categories of thunkable morphisms to the context of 2-categories. As is expected in the process of categorification, thunkability in this context will be a property of 2-cells but structure on 1-cells.

Definition 3.1. A *pseudomonad* on a 2-category $\mathcal{A}$ consists of a 2-functor $S : \mathcal{A} \rightarrow \mathcal{A}$, two pseudonatural transformations $\eta : 1_{\mathcal{A}} \Rightarrow S$, $\mu : S^2 \Rightarrow S$, and three invertible modifications $\lambda : \mu \cdot \eta_S \Rightarrow 1_S$, $\alpha : \mu \cdot S\mu \Rightarrow \mu \cdot \mu_S$ and $\rho : 1_S \Rightarrow \mu \cdot S\eta$, satisfying the coherences (1)-(5) as listed in Section 8 of [15]. A *pseudocomonad* is analogous, but with pseudonatural transformations $\varepsilon : S \Rightarrow 1_{\mathcal{A}}$ and $\delta : S \Rightarrow S^2$ in place of η and μ , respectively.

Definition 3.2. Let $\mathcal{B}$ be a 2-category. An abstract Kleisli structure (Q, θ) on $\mathcal{B}$ consists of

- A pseudocomonad $(Q, \varepsilon, \delta, \lambda, \alpha, \rho)$ on $\mathcal{B}$.
- A 2-functor $\theta : \mathcal{B}_0 \rightarrow \mathcal{B}^Q$ providing a lifting as in the following diagram, wherein $\mathcal{B}_0$ is the set of objects of $\mathcal{B}$ and $U^Q \dashv F^Q$ is the co-Eilenberg-Moore pseudoadjunction.

$$\begin{array}{ccccc}
\mathcal{B}_0 & \longrightarrow & \mathcal{B} & \xrightarrow{F^Q} & \mathcal{B}^Q \\
Q_0 \downarrow & & & \nearrow \theta & \downarrow U^Q \\
\mathcal{B}_0 & \longrightarrow & \mathcal{B} & &
\end{array}$$

The data of $\theta(X)$ will have its structure map written as $\theta_X : X \rightarrow QX$, counitor written as $u_X : 1_X \Rightarrow \varepsilon_X \cdot \theta_X$ and coassociator written as $m_X : \delta_X \cdot \theta_X \Rightarrow Q\theta_X \cdot \theta_X$.

Note that although we use the subscript X under θ , the assignment does not extend to a pseudonatural transformation. Similarly, nothing can be said about how u and m vary with X . However, the lifting condition says that $\theta_{QX} = \delta_X$, $u_{QX} = \rho_X$ and $m_{QX} = \alpha_X$.

Example 3.3. Let $(\mathcal{A}, S, \eta, \mu, \lambda, \alpha, \rho)$ be a pseudomonad. Then the 2-category of free pseudoalgebras and pseudomorphisms inherits an abstract Kleisli structure. The pseudocomonad is the one induced by the evident pseudoadjunction while the pseudocoalgebra associated to (SX, μ_X) has structure map given by $(S\eta_X, \mu_{\eta_X})$, counitor given by ρ_X and coassociator given by $S\eta_{\eta_X}$.

In Proposition 3.4, to follow, we give the construction of the 2-category of ‘morphisms equipped with thinkings, and thinkable 2-cells’ associated to an abstract Kleisli structure on a 2-category. As anticipated, thinkability is a property of a 2-cell but structure on a 1-cell.

Proposition 3.4. (Appendix 8.1) Let $\mathcal{B}$ be a 2-category equipped with an abstract Kleisli structure $(Q, \varepsilon, \delta, \theta)$, and let $\mathcal{B}_\theta$ denote the bijective on objects, 2-fully faithful factorisation of $\theta : \mathcal{B}_0 \rightarrow \mathcal{B}^Q$, as depicted below left. Then there is a pseudoadjunction as depicted below right in which F is bijective on objects and faithful on 2-cells. Moreover, the induced pseudocomonad on $\mathcal{B}$ is $(Q, \varepsilon, \delta, \lambda, \alpha, \rho)$.

$$\begin{array}{ccc}
\mathcal{B}_0 & \longrightarrow & \mathcal{B}_\theta \longrightarrow \mathcal{B}^Q \\
& & \mathcal{B} \xleftarrow[F]{\perp} \mathcal{B}_\theta \xrightarrow{U}
\end{array}$$

Definition 3.5. When (f, θ_f) is a morphism of $\mathcal{B}_\theta$, θ_f will be called a *thinking* of f . In this case f may be referred to as a *thunked morphism* and the pair (f, θ_f) will be called a *morphism equipped with a thinking*. The 2-cells in $\mathcal{B}_\theta$ will be called *thinkable*.

Proposition 3.6. Let $(\mathcal{B}, T)$ be a pseudomonad and consider the 2-category $(\mathcal{B}_T)_\theta$ formed by applying the construction of Proposition 3.4 on the abstract Kleisli structure described in Example 3.3. Then the left pseudoadjoint $\mathcal{B} \rightarrow \mathcal{B}_T$ factorises through the left pseudoadjoint $(\mathcal{B}_T)_\theta \rightarrow \mathcal{B}_T$ via a 2-functor $J : \mathcal{B} \rightarrow (\mathcal{B}_T)_\theta$.

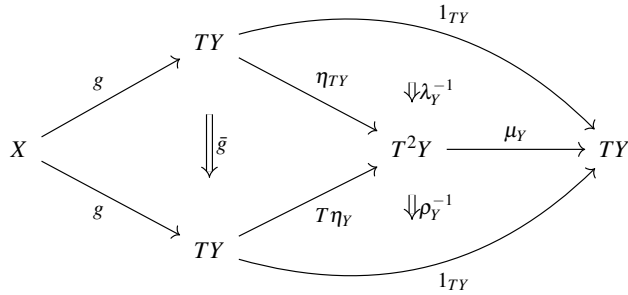
Proof. The morphism $J(f : X \rightarrow Y)$ has thinking given by the 2-cell $T\eta_f \in \mathcal{B}_T$. That this is well-defined as a 2-cell of pseudoalgebras follows from pseudonaturality of μ on η_f . That $T\eta_f$ does indeed equip (Tf, μ_f) with a well-defined thinking follows from the modification coherences for ρ and η_η on f . Finally, pseudonaturality of $T\eta$ on $\phi : f \Rightarrow g$ ensures that $J\phi$ is well-defined as a 2-cell in $(\mathcal{B}_T)_\theta$. $\square$

4 The isobidescent condition

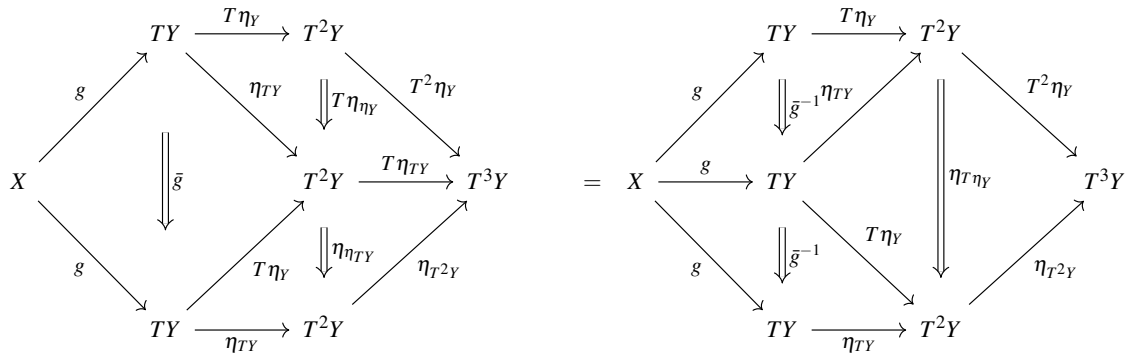
We will show in Theorem 6.8 that J is a suitable unit exhibiting abstract Kleisli structures on 2-categories as objects of a full reflective sub-**Gray**-category of $\mathbf{KLExt}(\mathbf{Gray})$. Before doing this we describe in Theorem 6.1 certain properties pseudomonads might have which are equivalent to J being a biequivalence. These conditions can be seen as categorifications of those in Theorem 2.9 from the setting of monads to the setting of pseudomonads. In the monads setting, one of these conditions is that η is the equaliser of $T\eta$ and η_T . As we will show, in the context of pseudomonads we also have a limit condition characterising those pseudomonads which correspond to abstract Kleisli structures on 2-categories. However, this equaliser condition is now replaced with the requirement that $(1_T, \eta, \eta_\eta)$ exhibit $1_{\mathcal{B}}$ as an isobidescent object. We recall this notion in the definition below.

Definition 4.1. Given a pseudomonad $(\mathcal{B}, T)$ and $X, Y \in \mathcal{B}$, define the category of *descent cones* from X to Y to have objects consisting of data of the form $(g, \bar{g})$ where

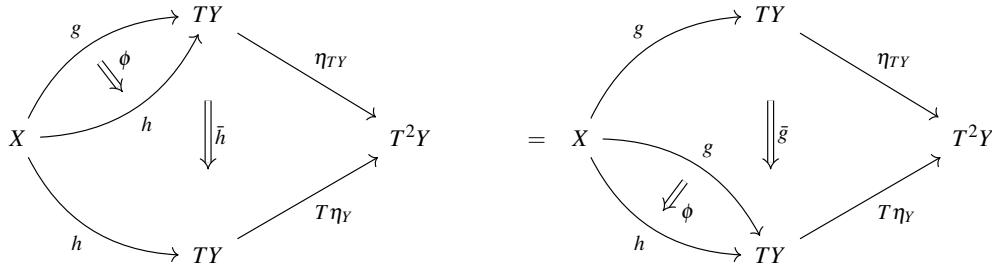
- $g : X \rightarrow TY$ is a 1-cell.
- $\bar{g} : \eta_{TY} \cdot g \Rightarrow T\eta_Y \cdot g$ is an invertible 2-cell.
- (Unit condition) The following pasting an identity.



- (Cocycle condition) The following equation holds.



Morphisms $\phi : (g, \bar{g}) \rightarrow (h, \bar{h})$ given by 2-cells $\phi : g \Rightarrow h$ in $\mathcal{A}$ satisfying the equation depicted below, with composition given by vertical composition in $\mathcal{B}$.



This category will be denoted as $\mathbf{Cone}_T(X, Y)$. We will say that $(\mathcal{B}, T)$ satisfies *isobidescent* if the canonical functor $(\eta \circ -, \eta_\eta \circ -) : \mathcal{B}(X, Y) \rightarrow \mathbf{Cone}_T(X, Y)$ which sends g to $(\eta_Y \cdot g, \eta_{\eta_Y} \cdot g)$ is an equivalence of categories.

Note that $(\mathcal{B}, T)$ satisfies the isobidescent condition precisely if for every object Y the data $(Y, \eta_Y, \eta_{\eta_Y})$ present Y as a bicategorical version of a descent object. Recall that bilimits have universal properties which hold up to pseudonatural biequivalence. This particular bilimit is dual to the one featuring in the monadicity theorem for pseudomonads [12].

5 Some intermediate results

We wish to show that the 2-functor J of Proposition 3.6 is a biequivalence if and only if $(\mathcal{B}, T)$ satisfies isobidescent. Our route towards this will be as follows.

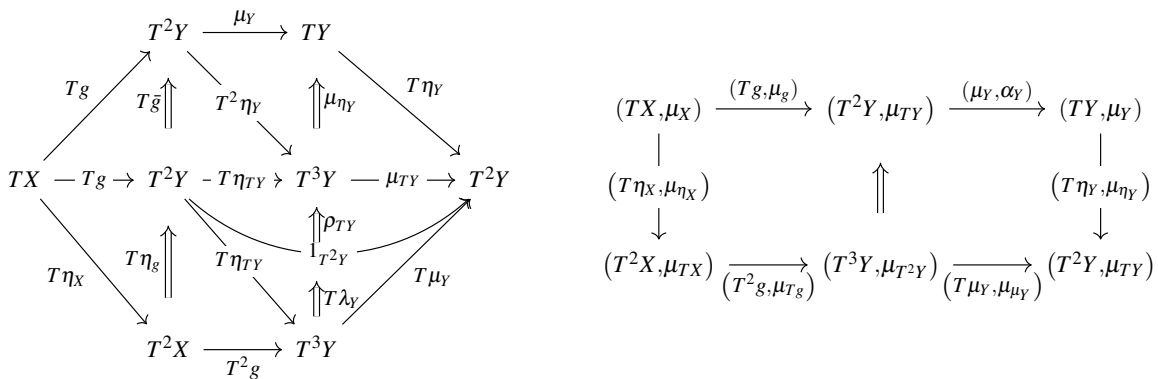
- In Proposition 5.2 we will describe functors displayed below.

$$\underline{J}_{X,Y} : \mathbf{Cone}_T(X, Y) \rightarrow (\mathcal{B}_T)_\theta((TX, \mu_X), (TY, \mu_Y))$$

- In Proposition 5.4 we will prove that these functors are equivalences.
- In Proposition 5.5 we will show that there are natural isomorphisms $\underline{J}_{X,Y} \cdot (\eta \circ -, \eta_\eta \circ -) \cong J_{X,Y}$.

The result will then follow in Theorem 6.1 by the two-out-of-three property for equivalences of categories. We begin by describing the thunking 2-cell of $\underline{J}_{X,Y}(g, \bar{g})$ and proving that it is a 2-cell of free pseudoalgebras.

Lemma 5.1. (Appendix 8.2) *Let $(\mathcal{B}, T)$ be a pseudomonad and $(g, \bar{g}) \in \mathbf{Cone}_T(X, Y)$. Then the pasting in the 2-category $\mathcal{B}$ depicted below left is a 2-cell of free T -pseudoalgebras as depicted below right.*



Lemma 5.3. (Appendix 8.4) *Let $((p, \bar{p}), \theta_p) : (TX, \mu_X) \rightarrow (TY, \mu_Y)$ be a morphism in $(\mathcal{B}_T)_\theta$. Then $p.\eta_X$ equipped with the following 2-cell defines a isobidescent cone from X to Y .*

Proposition 5.4. (Appendix 8.5) *The functor $\underline{J}_{X,Y} : \mathbf{Cone}_T(X,Y) \rightarrow (\mathcal{B}_T)_\theta((TX, \mu_X), (TY, \mu_Y))$ is an equivalence of categories.*

1. $\rho_Y : \underline{J}(\eta_Y g, \eta_{\eta_Y}) \rightarrow J(g)$ is a thunkable 2-cell of free pseudoalgebras.
2. ρ_Y is the component at g of a natural isomorphism

6 Main Results

Theorem 6.1. *Let $(\mathcal{B}, T)$ be a pseudomonad, $\mathcal{B}^T$ its Eilenberg-Moore object and $(\mathcal{B}^T)^{\bar{T}}$ the coEilenberg-Moore object of the induced pseudocomonad on $\mathcal{B}^T$. Similarly, let $(\mathcal{B}_T)^{\bar{T}}$ denote the coEilenberg-Moore object of the pseudocomonad induced on $\mathcal{B}_T$. Then the following are equivalent.*

1. $J: \mathcal{B} \rightarrow (\mathcal{B}_T)_\theta$ is a biequivalence.
2. $(\mathcal{B}, T)$ satisfies isobidescent.
3. The left pseudoadjoint $F_T: \mathcal{B} \rightarrow \mathcal{B}_T$ is faithful on 2-cells, full on thunkable 2-cells and surjective on 1-cells which admit a thunking.
4. The canonical comparison 2-functor $\underline{K}: \mathcal{B} \rightarrow (\mathcal{B}_T)^T$ is bi-fully-faithful.
5. The canonical comparison 2-functor $\bar{K}: \mathcal{B} \rightarrow (\mathcal{B}^T)^{\bar{T}}$ is bi-fully-faithful.

Proof. (1) $\iff$ (2) follows from Proposition 5.5 by the two-out-of-three property for equivalences of categories given that by Proposition 5.4 $\bar{G}_{X,Y}$ is an equivalence and G is bijective on objects. For (1) $\iff$ (3) observe $F_T = F_\theta \cdot J$ and F_θ is faithful on 2-cells, so this holds for F_T if and only if it holds for J . Meanwhile, F_T being full on thunkable 2-cells (resp. essentially surjective on 1-cells which admit a thunking) is clearly equivalent to J being full on 2-cells (resp. essentially surjective on 1-cells which admit a thunking), since these properties characterise the 1 and 2-cells which are in $(\mathcal{B}_T)_\theta$. The equivalence (2) $\iff$ (5) is true since by Proposition 3.4 part (7), the 2-category $(\mathcal{B}_T)_\theta$ is precisely the image of the canonical comparison $\underline{K} : \mathcal{B} \rightarrow (\mathcal{B}_T)^T$. Finally, for (4) $\iff$ (5) one observes that the image under $\bar{K} : \mathcal{B} \rightarrow (\mathcal{B}^T)^T$ of every $X \in \mathcal{B}$ will have an underlying T -pseudoalgebra which is free on X . Hence $\bar{K}$ will factorise through $\underline{K}$ via a 2-fully-faithful 2-functor, since $\mathcal{B}_T \rightarrow \mathcal{B}^T$ is 2-fully-faithful. $\square$

Definition 6.2. We describe a **Gray**-category that can be seen as categorifying $\mathbf{KL}(\mathbf{Cat})$, and whose description is dual of the description of $\mathbf{EM}(\mathcal{K})$ given in section 2.2 of [11]. It will be denoted as $\mathbf{KLExt}(\mathbf{Gray})_\lambda$, and has data as described below. There is also the **Gray**-category $\mathbf{KLExt}(\mathbf{Gray})_\tau$, described following Corollary 4.6 of [6], and an identity on objects and arrows **Gray**-functor $\mathbf{KLExt}(\mathbf{Gray})_\tau \rightarrow \mathbf{KLExt}(\mathbf{Gray})_\lambda$.

- Objects given by pseudomonads $(\mathcal{A}, S)$.
- Morphisms $(G, \bar{G}) : (\mathcal{A}, S) \rightarrow (\mathcal{B}, T)$ given by pairs of 2-functors $G : \mathcal{A} \rightarrow \mathcal{B}$ and $\bar{G} : \mathcal{A}_S \rightarrow \mathcal{B}_T$ satisfying $\bar{G} \cdot F_S = F_T \cdot G$, where F_S and F_T are the left pseudoadjoints to the 2-categories of free pseudoalgebras.
- 2-cells $\phi : (G, \bar{G}) \Rightarrow (H, \bar{H})$ given by arbitrary pseudonatural transformations $\phi : \bar{G} \Rightarrow \bar{H}$.
- 3-cells $\Omega : \phi \Rightarrow \psi$ are given by arbitrary modifications with source ϕ and target ψ .

We now turn to showing that J defines the unit of reflections from **Gray**-categories $\mathbf{KLExt}(\mathbf{Gray})_\kappa$, to **Gray**-categories of 2-abstract Kleisli structures.

Definition 6.3. Let 2-AbsKL_0 be the class of 2-abstract Kleisli structures. The **Gray**-category 2-AbsKL_τ will be defined as the intermediate **Gray**-category appearing in the bijective on objects/fully-faithful factorisation of the assignment $2\text{-AbsKL}_0 \rightarrow \mathbf{KLExt}(\mathbf{Gray})_\tau$ which sends the abstract Kleisli structure $(\mathcal{B}, T, \pi)$ to the pseudomonad $(\mathcal{B}_\pi, U_\pi \cdot F_\pi)$ induced by the pseudoadjunction described in Proposition 3.4 part 7. The **Gray**-category 2-AbsKL_λ will similarly be defined as the bijective on objects/fully-faithful factorisation of the same assignment this time viewed as $2\text{-AbsKL}_0 \rightarrow \mathbf{KLExt}(\mathbf{Gray})_\lambda$.

We will need Lemmas 6.4 and 6.6 to prove the desired universal property of $(1_{\mathcal{A}_S}, J)$. We fix the following notation

- $(A, S, \eta, \mu, \lambda, \alpha, \rho)$ is a pseudomonad.
- $((p, \bar{p}), \theta_p) : X \rightarrow Y$ is a 1-cell in $(\mathcal{A}_S)_\theta$.
- $(\mathcal{B}, T, \pi)$ is an abstract Kleisli structure on a 2-category $\mathcal{B}$.
- $(G, \bar{G})$ and $(H, \bar{H})$ are morphisms of 2-abstract Kleisli structures from $(\mathcal{A}_S, \underline{S}, \theta)$ to $(\mathcal{B}, T, \pi)$.
- $(\phi, \bar{\phi}) : (G, \bar{G}) \Rightarrow (H, \bar{H})$ and $(\psi, \bar{\psi}) : (G, \bar{G}) \Rightarrow (H, \bar{H})$ are tight 2-cells of 2-abstract Kleisli structures.
- $\Omega : (\phi, \bar{\phi}) \Rightarrow (\psi, \bar{\psi})$ is a tight 3-cell of 2-abstract Kleisli structures.

Lemma 6.4. *The morphism $\bar{G}(p, \bar{p})$ in $\mathcal{B}$ has a thunking given by the following pasting in $\mathcal{B}$.*

$$\begin{array}{ccccc}
 & GSX & \xrightarrow{\bar{G}(p, \bar{p})} & GSY & \\
 \pi_{GSX} \swarrow & \downarrow G\eta_X & \Downarrow \bar{G}\theta_p & \downarrow G\eta_Y & \searrow \pi_{GSY} \\
 & GS^2X & \xrightarrow{Gp} & GS^2Y & \\
 \pi_{G\eta_X} \swarrow & \downarrow \pi_{GS^2X} & \Downarrow \pi_{Gp} & \downarrow \pi_{GS^2X} & \searrow \pi_{G\eta_Y} \\
 TGSX & \xrightarrow{TG\eta_X} & TGS^2X & \xrightarrow{TGSp} & TGS^2Y & \xrightarrow{TG\eta_Y} & TGSY \\
 \swarrow T\bar{G}\rho_X^{-1} & \downarrow T\bar{G}(\mu_X, \alpha_X)_X & \Downarrow T\bar{G}\bar{p} & \downarrow T\bar{G}(\mu_Y, \alpha_Y)_Y & \swarrow T\bar{G}\rho_Y \\
 & TGSX & \xrightarrow{T\bar{G}(p, \bar{p})} & TGSY & \\
 1_{TGSX} \swarrow & & & & \searrow 1_{TGSY}
 \end{array}$$

Proof. This follows via similar techniques to those in other proofs. We omit details as they are significantly more tedious, but refer the interested reader to Appendix 11.5 and the proof of Lemma 8.2.14 of [16]. $\square$

Corollary 6.5.

1. *The assignment which sends the free pseudoalgebra (SX, μ_X) to GX , the thunked pseudomorphism $((p, \bar{p}), \theta_p) : (SX, \mu_X) \rightarrow (SY, \mu_Y)$ to $\bar{G}(p, \bar{p})$ equipped with the thunking given by the 2-cell described in Lemma 6.4, and the thunkable 2-cell $\chi : ((p, \bar{p}), \theta_p) \Rightarrow ((q, \bar{q}), \theta_q)$ to $\bar{G}\chi$ extends to a 2-functor $G' : (\mathcal{B}_T)_\theta \rightarrow \mathcal{B}_\pi$.*
2. *G' is the unique 2-functor satisfying*
 - (a) $G'J = G$, and
 - (b) $F_\pi \cdot G' = \bar{G} \cdot F_\theta$.

Proof. For part 1, first observe that if $\chi : ((p, \bar{p}), \theta_p) \Rightarrow ((q, \bar{q}), \theta_q)$ is a thunkable 2-cell in $\mathbf{FreePsAlg}_S$ then $\bar{G}\chi$ is also a thunkable 2-cell in $\mathcal{B}$. This follows from the thunkability condition for χ , pseudonaturality of π on $G\chi$, and the coherence for χ as a 2-cell of pseudoalgebras. Then functoriality of G' between hom categories is clear, while 2-functoriality of G' follows from that of $\bar{G}$, pseudonaturality of π , and by cancelling components of $T\bar{G}\rho$ and $\pi_{G\eta}$ with their inverses.

For part 2, G' satisfies condition (b) by construction. To see that it also satisfies condition (a), it suffices to consider the thunking described in Lemma 6.4 in the case where $((p, \bar{p}), \theta_p) = ((Sf, \mu_f), S\eta_f)$ and observe that this simplifies to π_{Gf} . This uses pseudonaturality of π on $G\eta_f$ and the modification coherence for ρ on f . Finally for uniqueness, observe that $\bar{G} \cdot F_\theta = F_\pi \cdot G''$ implies that $G''\chi = \bar{G}\chi$ for any 2-functor G'' , and that since G'' agrees with G' on 2-cells it must equal G' . $\square$

Lemma 6.6. *Let $(\phi, \bar{\phi}) : (G, \bar{G}) \Rightarrow (H, \bar{H})$ be a tight 2-cell of 2-abstract Kleisli structures. Then the pseudonaturality component $\bar{\phi}_{(p, \bar{p})}$ is a thunkable 2-cell in $\mathcal{B}$.*

Proof. This is proved using thunkability of ϕ_{η_X} and ϕ_{η_Y} , and pseudonaturality of $\bar{\phi}$, as detailed in Appendix 11.6 of [16]. $\square$

Corollary 6.7. *Let G' be as defined in 6.5 and let H' be defined analogously from $(H, \bar{H})$. Then*

1. *There is a pseudonatural transformation $\phi' : G' \Rightarrow H'$ with component at X given by ϕ_X and component at $((p, \bar{p}), \theta_p)$ given by $\bar{\phi}_{(p, \bar{p})}$.*
2. *ϕ' is the unique pseudonatural transformation satisfying*
 - (a) *$\phi'J = \phi$, and*
 - (b) *$\bar{\phi}.F_\theta = F_\pi.\phi'$.*

Proof. For part (1), the conditions for pseudonaturality of ϕ' follow directly from the analogous conditions for $\bar{\phi}$. For part (2) it is clear from the definition of ϕ' and the fact that $F_\pi.\phi = \bar{\phi}.F_\theta$ that ϕ' uniquely satisfies conditions (a) and (b). $\square$

Theorem 6.8. *The inclusion $I : 2\text{-}\mathbf{AbsKL} \rightarrow \mathbf{KLExt}(\mathbf{Gray})$ has a left $\mathbf{Gray}$ -adjoint which sends $(\mathcal{A}, S)$ to the 2-abstract Kleisli structure $(\mathcal{A}_S, \underline{S}, F_S\eta)$, and the unit of this adjunction at $(\mathcal{A}, S)$ is given by $(J, 1_{\mathcal{A}_S}) : (\mathcal{A}, S) \rightarrow ((\mathcal{A}_S)_\theta, S')$.*

Proof. It suffices to show that the 2-functor depicted below, which is induced by precomposition along $(J, 1_{\mathbf{FreePsAlgs}})$, is an isomorphism of 2-categories.

$$\mathbf{KLExt}(\mathbf{Gray})((\mathcal{A}_S)_\theta, \underline{S}'), (\mathcal{B}_\pi, T')) \rightarrow \mathbf{KLExt}(\mathbf{Gray})((\mathcal{A}, S), (\mathcal{B}_\pi, T'))$$

In Corollaries 6.5 and 6.7 we have already seen that the actions of this 2-functor on objects and on morphisms are bijections. Let $(\Omega, \bar{\Omega}) : (\phi, \bar{\phi}) \Rightarrow (\psi, \bar{\psi})$ be a 3-cell of 2-abstract Kleisli structures. Then observe that $\bar{\Omega}_X = \Omega_X$ for every $X \in \mathcal{A}$, and hence the modification coherence for $\bar{\Omega}$ implies that $X \mapsto \Omega_X$ also extends to a modification $\Omega' : \phi' \Rightarrow \psi'$. Finally, observe that Ω' is indeed the unique modification $\phi' \Rightarrow \psi'$ satisfying $\Omega'J = \Omega$ and $F_\pi.\Omega = \bar{\Omega}.F_\theta$. All of these observations are straightforward since the 2-functors F_π, F_θ and J are all bijective on objects. This completes the proof. $\square$

7 Concluding remarks

Führmann showed that abstract Kleisli structures form a full reflective sub-category of the category of monads and strict morphisms, whose essential image consists of those monads whose unit $\eta : 1_{\mathcal{A}} \Rightarrow S$ is the equaliser of $S\eta$ and η_S . We have further abstracted these structures, and shown that Führmann's adjunction underlies a reflective 2-adjunction. We then defined abstract Kleisli structures on 2-categories, and proven two-dimensional analogues of both Führmann's results and our extensions of those results. Specifically, we have shown that abstract Kleisli structures on 2-categories can be seen as the objects of full sub- $\mathbf{Gray}$ -categories of either $\mathbf{KLExt}(\mathbf{Gray})_\tau$ or $\mathbf{KLExt}(\mathbf{Gray})_\lambda$, with these structures being described in following Corollary 4.6 in [6] and in Definition 6.2, respectively. In both instances, there is a reflection to the inclusion given by passing to the 2-category whose morphisms are equipped with thunkings, and whose 2-cells are thunkable. If the data (η, η_η) extracted from a given pseudomonad is a certain isobidescent cone, then that pseudomonad is biequivalent to the canonical pseudomonad formed by an abstract Kleisli structure on a 2-category. The base of this pseudomonad consists of morphisms equipped with thunkings, and the thunkable 2-cells. Also equivalent to this bicategorical limit condition are certain criteria on the left pseudoajoint, or bi-fully faithfulness of comparisons from the base 2-category to 2-categories of descent data.

8 Appendices of proofs

Notation 8.1. We use colour to draw the reader's attention to new data appearing in each step of a proof involving a pasting diagram chase. In particular, we use *blue* for new objects and morphisms and *red* for new 2-cells. To avoid clutter, we omit denoting inverses of 2-cells with $(-)^{-1}$. The reader should be able to infer from the source and target of an invertible 2-cell denoted γ if it is actually the inverse γ^{-1} .

8.1 Proof of Proposition 3.4

We first observe that the 2-category $\mathcal{B}_\theta$ has

- Objects the same as $\mathcal{B}$.
- Arrows $(f, \theta_f) : X \rightarrow Y$ consisting of an arrow $f : X \rightarrow Y$ in $\mathcal{B}$ and an invertible 2-cell $\theta_f : \theta_Y \cdot f \Rightarrow Qf \cdot \theta_X$ satisfying the following equations.

– (Unit condition)

$$\begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \theta_X \downarrow & \Downarrow \theta_f & \downarrow \theta_Y \\
 QX & \xrightarrow{Qf} & QY \\
 & \xrightarrow{\varepsilon_Y} & Y
 \end{array}
 \quad
 \begin{array}{ccc}
 X & \xrightarrow{1_X} & X \\
 \theta_X \downarrow & \searrow \varepsilon_X & \downarrow \varepsilon_f \\
 QX & \xrightarrow{Qf} & QY \\
 & \xrightarrow{\varepsilon_Y} & Y
 \end{array}$$

– (Associativity condition)

$$\begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \theta_X \searrow & \Downarrow \theta_f & \swarrow \theta_Y \\
 QX & \xrightarrow{Qf} & QY \\
 Q\theta_X \searrow & \Downarrow \delta_f & \swarrow \delta_Y \\
 Q^2X & \xrightarrow{Q^2f} & Q^2Y
 \end{array}
 \quad
 \begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \theta_X \searrow & \Downarrow \theta_f & \swarrow \theta_Y \\
 QX & \xrightarrow{Qf} & QY \\
 Q\theta_X \searrow & \Downarrow Q\theta_f & \swarrow Q\theta_Y \\
 Q^2X & \xrightarrow{Q^2f} & Q^2Y
 \end{array}$$

- 2-cells $\phi : (f, \theta_f) \rightarrow (g, \theta_g)$ given by 2-cells $\phi : f \Rightarrow g$ in $\mathcal{B}$ satisfying the following equation.

$$\begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \theta_X \downarrow & \Downarrow \phi & \downarrow \theta_Y \\
 QX & \xrightarrow{Qf} & QY \\
 & \xrightarrow{Qg} &
 \end{array}
 \quad
 \begin{array}{ccc}
 X & \xrightarrow{f} & Y \\
 \theta_X \downarrow & \Downarrow \theta_f & \downarrow \theta_Y \\
 QX & \xrightarrow{Qf} & QY \\
 & \xrightarrow{Qg} &
 \end{array}$$

There is a 2-functor $F : \mathcal{B}_\theta \rightarrow \mathcal{B}$ which forgets data of the form θ_f , and is hence clearly bijective on objects and faithful on 2-cells. Moreover, there is a 2-functor $U : \mathcal{B} \rightarrow \mathcal{B}_\theta$ which sends 1-cells f to (Qf, δ_f) and takes the image under Q on objects and 2-cells. It follows from the equations displayed

8.2 Proof of Lemma 5.1

The figure consists of two commutative diagrams, labeled (a) and (b), illustrating the relationship between the monoidal product and the tensor product of monoids.

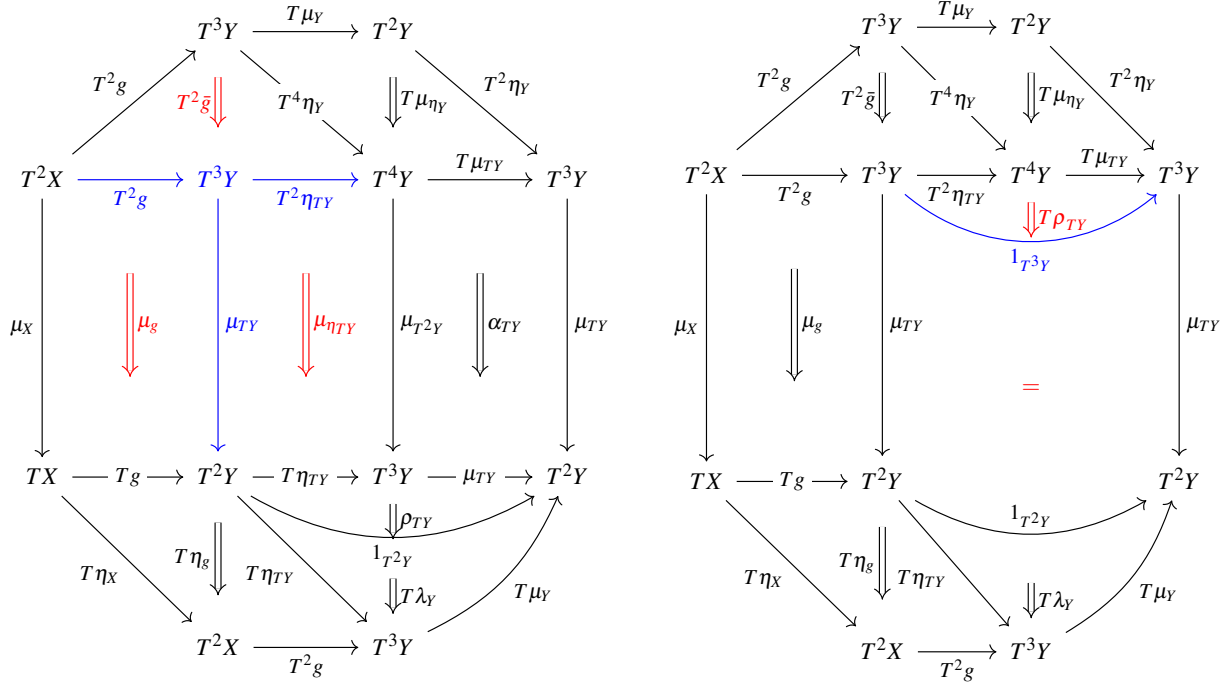
Diagram (a): Monoidal Product of Monoids

This diagram shows the monoidal product of monoids. The objects are arranged in a grid-like structure. The top row consists of T^3Y , T^2Y , and T^3Y . The middle row consists of T^2X , T^2Y , and T^3Y . The bottom row consists of TX , T^2Y , T^3Y , and T^2Y . Arrows represent multiplication (μ), comultiplication (η), and the monoidal product (T). For example, $T^2X \xrightarrow{\mu_X} TX$, $TX \xrightarrow{Tg} T^2Y$, $T^2Y \xrightarrow{\mu_Y} TY$, $TY \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2X \xrightarrow{T\eta_X} T^2Y$, $T^2Y \xrightarrow{T\eta_{TY}} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2X \xrightarrow{T^2g} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$.

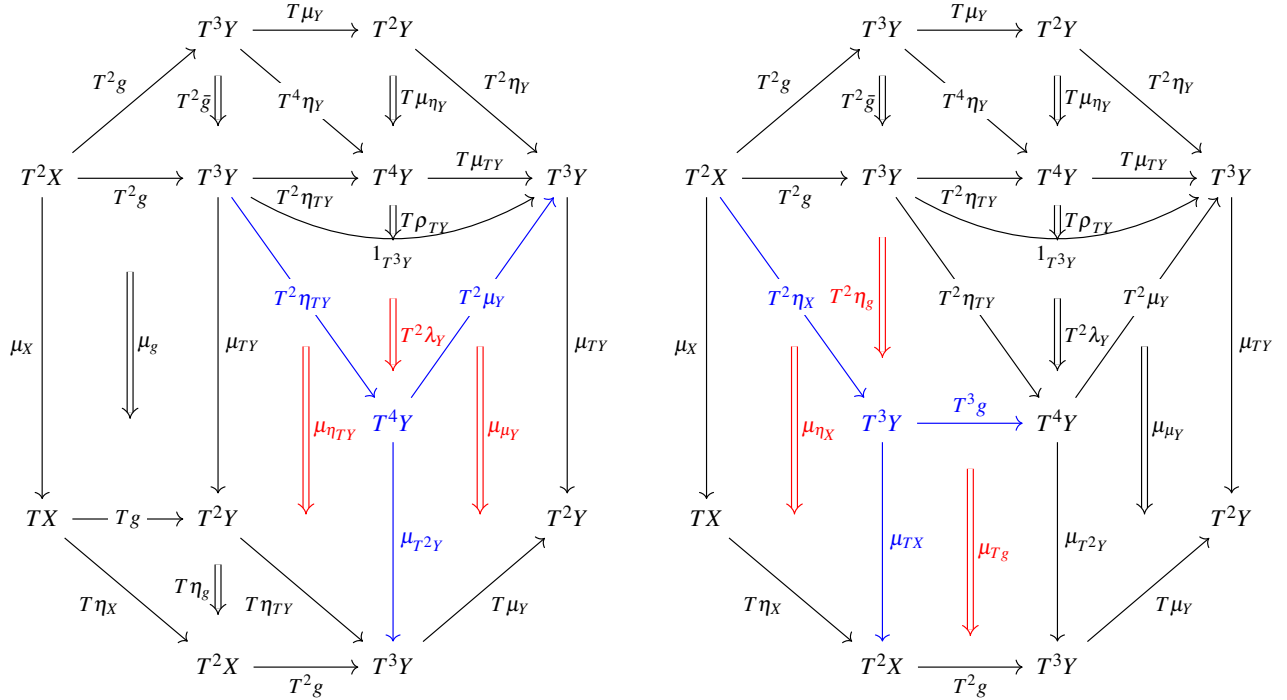
Diagram (b): Tensor Product of Monoids

This diagram shows the tensor product of monoids. The objects are arranged in a grid-like structure. The top row consists of T^3Y , T^2Y , and T^3Y . The middle row consists of T^2X , T^2Y , and T^3Y . The bottom row consists of TX , T^2Y , T^3Y , and T^2Y . Arrows represent multiplication (μ), comultiplication (η), and the tensor product ($\otimes$). For example, $T^2X \xrightarrow{\mu_X} TX$, $TX \xrightarrow{Tg} T^2Y$, $T^2Y \xrightarrow{\mu_Y} TY$, $TY \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2X \xrightarrow{T\eta_X} T^2Y$, $T^2Y \xrightarrow{T\eta_{TY}} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2X \xrightarrow{T^2g} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$, $T^2Y \xrightarrow{T\eta_Y} T^3Y$, $T^3Y \xrightarrow{\mu_{TY}} T^2Y$.

Then apply pseudonaturality of μ on $\bar{g}$ to reduce above right to below left, followed by coherence 5 for the pseudomonad (A, T) to reduce below left to below right.

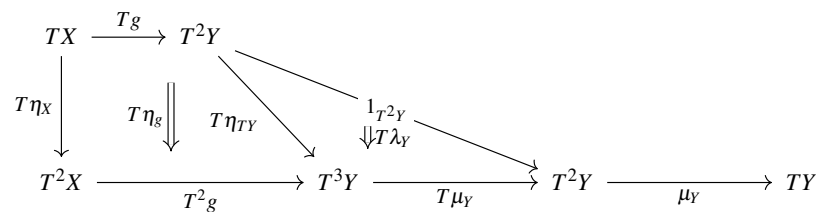
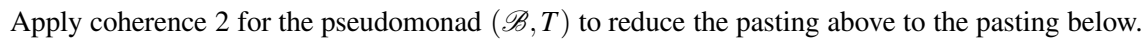
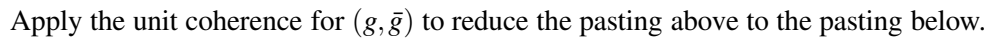
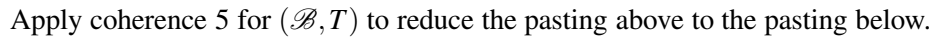


Apply pseudonaturality of μ on λ_Y to reduce above right to below left. Finally, apply pseudonaturality of μ on η_g to reduce below left to below right and complete the proof.



□

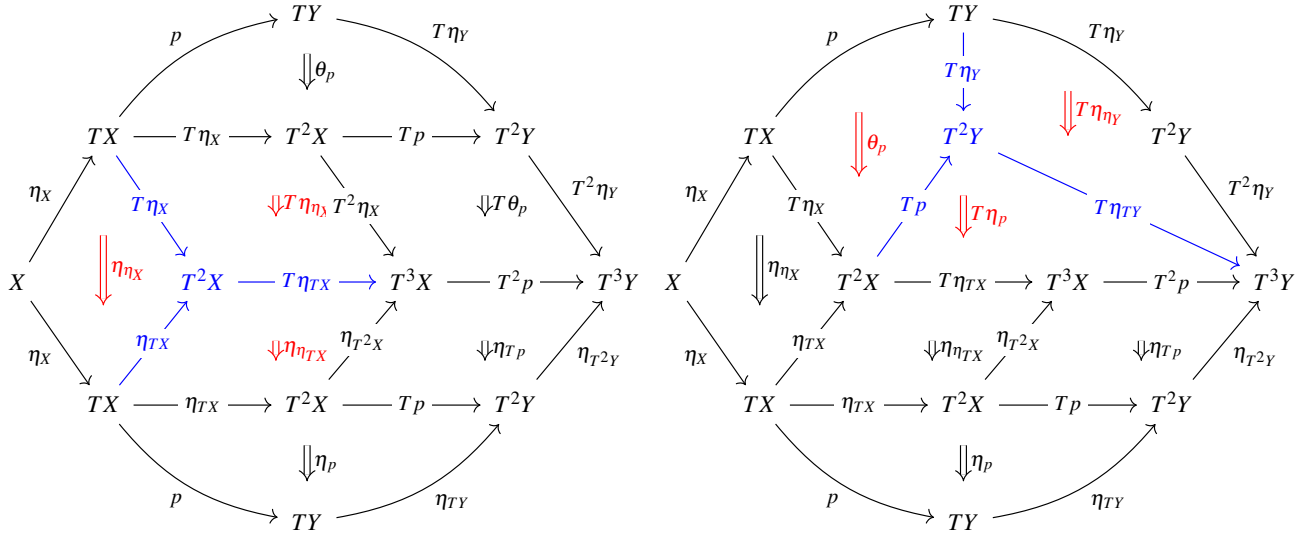
Proof. We need to verify that $\underline{J}(g, \bar{g})$ is well-defined as a morphism equipped with a thunking. For the unit condition, begin with the pasting below.



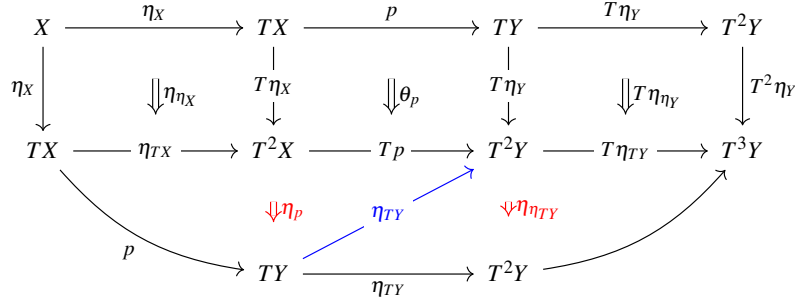
$$\begin{array}{ccccccc}
TX & \xrightarrow{Tg} & T^2Y & \xrightarrow{l_{T^2Y}} & T^2Y & & \\
T\eta_X \downarrow & & T\eta_g \Downarrow & T\eta_{TY} \searrow & \downarrow \rho_{TY} & \nearrow \mu_{TY} & \downarrow \alpha_Y \\
T^2X & \xrightarrow{T^2g} & T^3Y & \xrightarrow{T\mu_Y} & T^2Y & \xrightarrow{\mu_Y} & TY
\end{array}$$
$$\begin{array}{ccccccc}
TX & \xrightarrow{T\eta_X} & T^2X & \xrightarrow{T^2g} & T^3Y & \xrightarrow{T\mu_Y} & T^2Y \\
& \searrow 1_{TX} & \downarrow \mu_X & & \downarrow \mu_{TY} & \Downarrow \alpha_Y & \downarrow \mu_Y \\
& & TX & \xrightarrow{Tg} & T^2Y & \xrightarrow{\mu_Y} & TY
\end{array}$$

The figure consists of two commutative diagrams. The left diagram shows the naturality of the tangent functor T . It features objects X , TX , TY , T^2X , T^2Y , and T^3Y . Morphisms include η_X , η_{TX} , η_{TY} , η_{T^2X} , η_{T^2Y} , η_{T^3Y} , $T\eta_X$, $T\eta_Y$, $T^2\eta_X$, $T^2\eta_Y$, Tp , and T^2p . The right diagram shows the naturality of the second-order tangent functor T^2 . It features objects X , TX , TY , T^2X , T^2Y , T^3X , and T^3Y . Morphisms include η_X , η_{TX} , η_{TY} , η_{T^2X} , η_{T^2Y} , η_{T^3Y} , $T\eta_X$, $T\eta_Y$, $T^2\eta_X$, $T^2\eta_Y$, Tp , T^2p , and T^3p . The right diagram uses blue and red colors to highlight specific morphisms and their relationships.

Apply pseudonaturality of η on η_X to reduce the pasting above right to the pasting below left. Apply the associativity coherence for $((p, \bar{p}), \theta_p)$ as a 1-cell in $(\mathcal{B}_T)_\theta$ to reduce the pasting below left to the pasting below right.



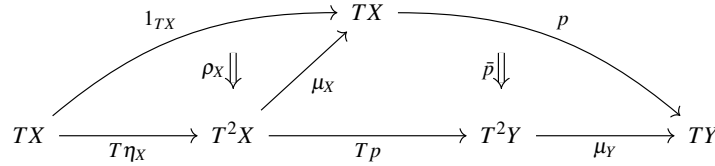
Finally, apply pseudonaturality of η on p to reduce the pasting above right to the pasting below, and observe that this completes the proof.



□

8.5 Proof of Proposition 5.4

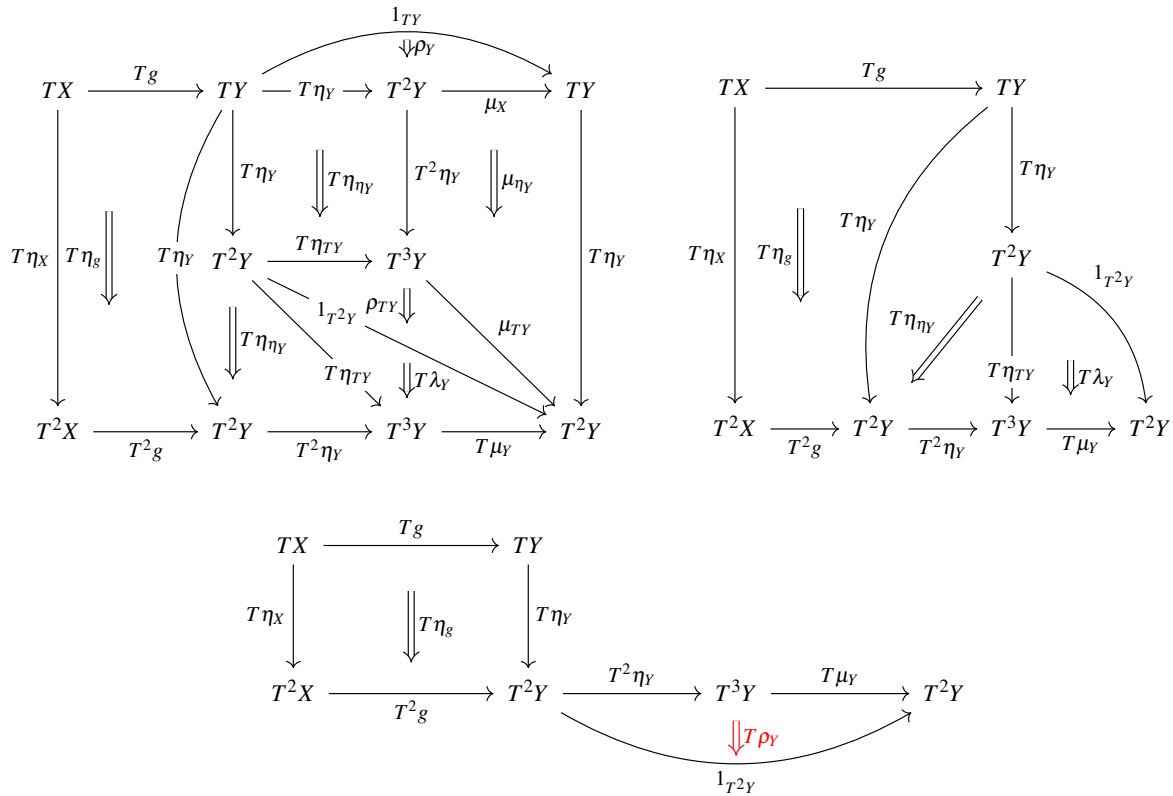
For essential surjectivity on objects, we claim that the image under $\underline{J}_{X,Y}$ of the cone from X to Y defined in Lemma 5.3 is isomorphic to $((p, \bar{p}), \theta_p)$, via the 2-cell of pseudoalgebras depicted below.



Thunkability of this 2-cell follows from a diagram chase given in Appendix 11.4 of [16]. For fully faithfulness, we already know that arbitrary 2-cells ϕ from $g : X \rightarrow TY$ to $h : X \rightarrow TY$ are in bijection with 2-cells of free pseudoalgebras $(\mu_Y, \alpha_Y) \cdot (Tg, \mu_g) \Rightarrow (\mu_Y, \alpha_Y) \cdot (Th, \mu_h)$. This is part of the biequivalence between the Kleisli bicategory of (A, T) and $\mathcal{B}_T$. It therefore suffices to observe that if $\phi : ((p, \bar{p}), \theta_p) \Rightarrow ((q, \bar{q}), \theta_q)$ is thunkable then $\psi \cdot \eta_X$ is a morphism in $\mathbf{Cone}_T(X, Y)$. This follows from pseudonaturality of η on ϕ .

8.6 Proof of Proposition 5.5

For part (1), begin with the pasting depicted below top left and apply the modification coherence for ρ on η_Y to reduce to the pasting depicted below top right. Finally, apply coherence 3 for $(\mathcal{B}, T)$ to arrive at the pasting depicted below and observe that this completes the proof. For part (2), naturality follows by middle-four interchange.



References

- [1] Barr, M., Wells, C., Toposes, triples and theories, *Reprints in Theory and Applications of Categories*, No. 12, pp. 1-287, (2005)
- [2] Cruttwell, G., Gavranović, B., Ghani, N., Wilson, P., Zanasi, F. Categorical Foundations of Gradient-Based Learning, *Programming Languages and Systems*, 31st European Symposium on Programming, (2022) https://dx.doi.org/10.1007/978-3-030-99336-8_1
- [3] Erwig, M., Ren, D., Monadification of Functional Programs, *Science of Computer Programming*, Vol. 52, Issues 1–3, pp. 101-129, (2004) <https://doi.org/10.1016/j.scico.2004.03.004>
- [4] Fiore, M., Gambino, N., Hyland, M., Monoidal Bicategories, Differential Linear Logic, and Analytic Functors, <https://doi.org/10.48550/arXiv.2405.05774>
- [5] Führmann, C. Direct models of the computational λ -calculus, *Electronic Notes in Theoretical Computer Science*, Vol. 20, pp 245-292 (1999), [https://doi.org/10.1016/S1571-0661\(04\)80078-1](https://doi.org/10.1016/S1571-0661(04)80078-1)
- [6] Gambino, N., Lobbia, G. On the formal theory of pseudomonads and pseudodistributive laws, *Theory and Applications of Categories*, Vol. 37, No. 2, pp 14-56, (2021)

- [7] Gray, J. W., *Formal Category Theory: Adjointness for 2-categories*, Springer Lecture Notes in Mathematics, Vol. 391, (1974), <https://doi.org/10.1007/BFb0061280>
- [8] Gurski, N., *Coherence in Three-Dimensional Category Theory*, Cambridge Tracts in Mathematics, Vol. 201, (2013), <https://doi.org/10.1017/CB09781139542333>
- [9] Kelly, G. M., Elementary Observations on 2-Categorical Limits, *Bulletin of the Australian Mathematical Society*, Vol. 39, pp. 301-317, (1989), <https://doi.org/10.1017/S0004972700002781>
- [10] Lack, S., Miranda, A., What is the universal property of the 2-category of monads?, *Theory and Applications of Categories*, Vol. 42, 2024, No. 1, pp 2-18., (2024)
- [11] Lack, S., Street, S., The Formal Theory of Monads II, *Journal of Pure and Applied Algebra*, Vol. 175, pp 243-265, (2002) [https://doi.org/10.1016/S0022-4049\(02\)00137-8](https://doi.org/10.1016/S0022-4049(02)00137-8)
- [12] Le Creurer, I.J., Marmolejo, F, Vitale, E.M.. Beck's theorem for pseudo-monads. *Journal of Pure and Applied Algebra*, Vol. 173, pp 293-313, (2000), [https://doi.org/10.1016/S0022-4049\(02\)00038-5](https://doi.org/10.1016/S0022-4049(02)00038-5)
- [13] Lemay, J. S., Cartesian Differential Kleisli Categories, *Electronic Notes in Theoretical Informatics and Computer Science*, Vol. 3, Proceedings of Mathematical Foundations of Programming Semantics XXXIX, (2023), <https://doi.org/10.46298/entics.12278>
- [14] Marmolejo, F., Distributive laws for pseudomonads I, *Theory and Applications of Categories*, Vol. 5, No. 5, pp 81-147, (1999).
- [15] Marmolejo, F., Doctrines whose structure forms a fully faithful adjoint string, *Theory and Applications of Categories*, Vol. 3, No. 2, pp. 22-23, (1997).
- [16] Miranda, A., Topics in Low Dimensional Higher Category Theory, Ph. D. Thesis, Macquarie University, (2023)
- [17] Paquet, H., Saville, P., Strong Pseudomonads and Premonoidal Bicategories, available at <https://arxiv.org/abs/2304.11014>
- [18] Power, A. J., A 2-categorical pasting theorem, *Journal of Algebra*, Volume 129, Issue 2, pp. 439-445, (1990), [https://doi.org/10.1016/0021-8693\(90\)90229-H](https://doi.org/10.1016/0021-8693(90)90229-H)
- [19] Street R. The formal theory of monads, *Journal of Pure and Applied Algebra*, Vol. 2, pp. 149-168, (1978), [https://doi.org/10.1016/0022-4049\(72\)90019-9](https://doi.org/10.1016/0022-4049(72)90019-9)

Learners are Almost Free Compact Closed

Mitchell Riley

Mathematics, Division of Science

New York University Abu Dhabi

`mitchell.v.riley@nyu.edu`

The category of learners has a pleasant symmetric formulation when the morphisms are considered up to a coarser equivalence than the one originally described in the paper "Backprop as Functor". A quotient of this modified category gives a new construction of the free compact closed category on a symmetric monoidal category.

In this note I expand on an observation made at the end of [15] and further discussed online [16]. If the definition of a learner [7] is tweaked slightly so that learners are considered identical under a coarser equivalence relation than before, a striking symmetry in the definition is revealed.

This new category comes quite close to being compact closed. There is a strict symmetric monoidal involution and each object has a cup and cap morphism connecting it to its “dual” under this involution. However, the snake equations fail and so these are not true duals. Quotienting the category to force the equations to hold results in a compact closed category, and we end the note by showing that it is in fact the *free* compact closed category on the starting symmetric monoidal category.

Compact closed categories support a more powerful diagrammatic calculus than bare symmetric monoidal categories, so this work may make it easier to design and reason about learners compositionally. The operation that dualises a learner does not appear to have been previously noticed and there is much to investigate. As asked by David Spivak, what is the dual of a neuron?

1 Extensional Learners

Learners are a categorical construction used in a compositional approach to supervised learning and backpropagation [7, 2]. We first recall their definition.

Definition 1.1 ([7, Definition 2.1]). For sets A and B , a *learner* $A \rightarrow B$ is a “parameter” set P together with a triple of functions $I : P \times A \rightarrow B$ and $U : P \times A \times B \rightarrow P$ and $r : P \times A \times B \rightarrow A$, considered up to isomorphism of parameter sets. That is, two learners (P, I, U, r) and (P', I', U', r') are identified whenever there is a bijection $f : P \rightarrow P'$ with

$$\begin{aligned} I'(f(p), a) &= I(p, a) \\ U'(f(p), a, b) &= f(U(p, a, b)) \\ r'(f(p), a, b) &= r(p, a, b) \end{aligned}$$

Learners expose their parameter sets P to the outside world (up to isomorphism), and two learners with non-isomorphic parameter sets are considered different even if they ‘behave the same’ on all input data A and B . Cribbing some terminology from type theory, we will label these learners *intensional*: their identity depends on the specific choice of parameter set even if it is not observable through their

The author acknowledges support by *Tamkeen* under *NYUAD Research Institute grant CG008*.

input-output behaviour. (Unfortunately this clashes with the meaning of intensional/extensional as used in [10].)

The equivalence relation on learners has a clear generalisation to any category with finite products, and in this generality:

Proposition 1.2 ([7, Proposition 2.4]). *Intensional learners in any category with finite products form a symmetric monoidal category $\mathbf{IntLearn}_{\mathcal{C}}$.* $\square$

The data underlying a learner $A \rightarrowtail B$ with fixed parameter set P has an appealing symmetric description, via the chain of isomorphisms

$$\begin{aligned} & \mathcal{C}(P \times A, B) \times \mathcal{C}(P \times A \times B, P) \times \mathcal{C}(P \times A \times B, A) \\ & \cong \mathcal{C}(P \times A, B) \times \mathcal{C}(P \times A \times B, P \times A) \\ & \cong \int^{Q:\mathcal{C}} \mathcal{C}(P \times A, Q) \times \mathcal{C}(P \times A, B) \times \mathcal{C}(Q \times B, P \times A) \\ & \cong \int^{Q:\mathcal{C}} \mathcal{C}(P \times A, Q \times B) \times \mathcal{C}(Q \times B, P \times A) \end{aligned}$$

The quotient up to isomorphism of P can then be expressed as an additional coend over the core of $\mathcal{C}$:

$$\mathbf{IntLearn}(A, B) = \int^{P:\mathbf{Core}(\mathcal{C})} \int^{Q:\mathcal{C}} \mathcal{C}(P \times A, Q \times B) \times \mathcal{C}(Q \times B, P \times A)$$

Three generalisations immediately suggest themselves:

- replace the former coend with one over all of $\mathcal{C}$ rather than only the isomorphisms,
- replace $\times$ with an arbitrary symmetric monoidal product $\otimes$, and,
- allow the repeat occurrences of A and B to be different objects, as in the category of optics [1, 15].

We arrive at our generalised definition of learner.

Definition 1.3. For objects A, A', B, B' of a symmetric monoidal category $\mathcal{C}$, an *extensional learner* $(A, A') \rightarrowtail (B, B')$ is an element of

$$\mathbf{Learn}_{\mathcal{C}}((A, A'), (B, B')) := \int^{P, Q:\mathcal{C}} \mathcal{C}(P \otimes A, Q \otimes B) \times \mathcal{C}(Q \otimes B', P \otimes A')$$

The extensional learner represented by a pair of maps $f : P \otimes A \rightarrow Q \otimes B$ and $g : Q \otimes B' \rightarrow P \otimes A'$ will be written $(f \mid g) : (A, A') \rightarrowtail (B, B')$.

Because these are our focus, we omit ‘extensional’ and call these simply *learners*.

When specialised back to the case of sets and binary products, this definition unwinds to a coarser notion of equivalence between a pair of intensional learners. Two learners (P, I, U, r) and (P', I', U', r') are identified by this new definition when there is a (not-necessarily-bijective) function $f : P \rightarrow P'$ and an intermediate function $\hat{U} : P' \times A \times B \rightarrow P$ such that

$$\begin{aligned} I'(f(p), a) &= I(p, a) \\ r'(f(p), a, b) &= r(p, a, b) \\ \hat{U}(f(p), a, b) &= U(p, a, b) \\ U'(p', a, b) &= f(\hat{U}(p', a, b)) \end{aligned}$$

That is, the relationship of f to I and r is the same as before, but for U we require a diagonal filler in the diagram

$$\begin{array}{ccc}
P \times A \times B & \xrightarrow{U} & P \\
f \times A \times B \downarrow & \hat{U} \nearrow & \downarrow f \\
P' \times A \times B & \xrightarrow{U'} & P'
\end{array}$$

Lemma 1.4. *Intensionally equivalent learners are extensionally equivalent.*

Proof. The bijection $f : P \rightarrow P'$ satisfies the first two equations of extensional equivalence by assumption. For the intermediate function $\hat{U} : P' \times A \times B \rightarrow P$, take $\hat{U}(p', a, b) := U(f^{-1}(p'), a, b)$, then

$$\begin{aligned}
\hat{U}(f(p), a, b) &= U(f^{-1}(f(p)), a, b) = U(p, a, b) \\
f(\hat{U}(p', a, b)) &= f(U(f^{-1}(p'), a, b)) = U'(f(f^{-1}(p')), a, b) = U'(p', a, b)
\end{aligned}$$

□

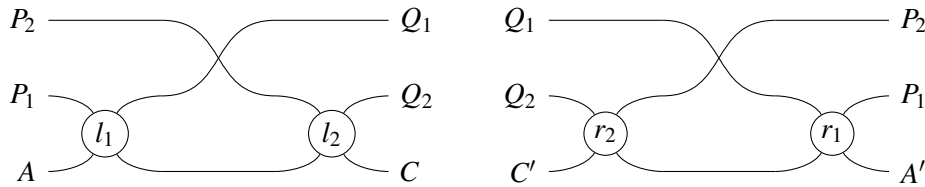
Remark 1.5. The extensional relation is similar to the notion of 2-morphism between learners given in [7, Definition 7.1], which only requires that the outer square in the above diagram commutes. The intensional relation identifies two learners when there is a 2-isomorphism between them.

In [6], Fong and Johnson consider two intensional learners identical when there is a 2-morphism such that f is only *surjective*, and not necessarily a bijection. This is a coarser notion of equivalence still: for example, it identifies all intensional learners $1 \rightarrow 1$.

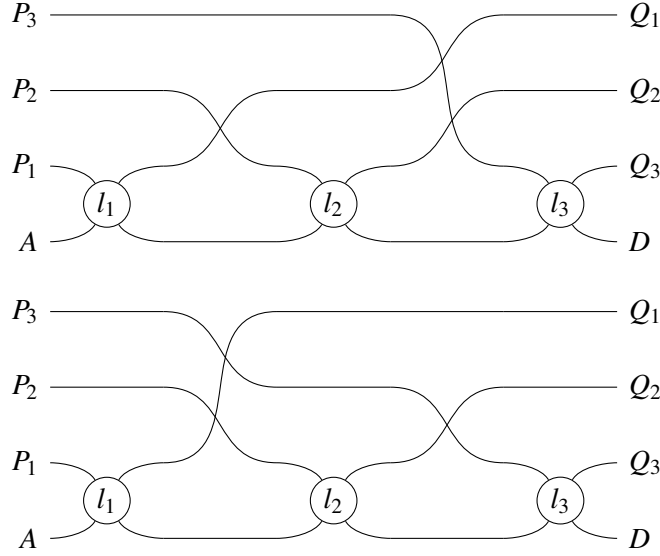
Proposition 1.6. *Learners form a category $\mathbf{Learn}_{\mathcal{C}}$, where the identity learner $(A, A') \rightarrow (A, A')$ is $(\text{id}_{I \otimes A} \mid \text{id}_{I \otimes A'})$, and the composite of $(l_1 \mid r_1) : (A, A') \rightarrow (B, B')$ and $(l_2 \mid r_2) : (B, B') \rightarrow (C, C')$ with representatives*

$$\begin{aligned}
l_1 &: P_1 \otimes A \rightarrow Q_1 \otimes B \\
r_1 &: Q_1 \otimes B' \rightarrow P_1 \otimes A' \\
l_2 &: P_2 \otimes B \rightarrow Q_2 \otimes C \\
r_2 &: Q_2 \otimes C' \rightarrow P_2 \otimes B'
\end{aligned}$$

is given by



Proof. Similar to [15, Proposition 2.0.3]. The unit laws for composition follow quickly from the unit laws in $\mathcal{C}$. Associativity of composition follows (for the left component) from the isotopy of the following string diagrams together with associativity of $\mathcal{C}$.



The right component is symmetric. The coend relation ensures that the associativity morphisms in $\mathcal{C}$ for the triples (P_1, P_2, P_3) and (Q_1, Q_2, Q_3) can be used without changing the identity of the learner, as each such morphism is paired with its inverse across the gap. $\square$

Remark 1.7. Intensional learners manifestly do not form a locally small category: any choice of parameter set P determines a learner $(1, 1) \rightarrow (1, 1)$ with U given by projection, and any two sets with different cardinalities determine two intensionally inequivalent learners. The situation is not improved with the extensional relation: these learners remain unequal and $\mathbf{Learn}_{\mathcal{C}}$ is not locally small. (Compare with ordinary lenses, which can be described in a similar coend style but always form a locally small category.) For a symmetric monoidal category, the collection of learners $(I, I) \rightarrow (I, I)$ is the *trace* of $\mathcal{C}$ in the sense of [4], which can be difficult to calculate even in simple cases.

The category $\mathbf{Learn}_{\mathcal{C}}$ accepts an obvious functor $\iota : \mathcal{C} \rightarrow \mathbf{Learn}_{\mathcal{C}}$, sending $f : A \rightarrow B$ to $(I \otimes f \mid \text{id}_{I \otimes I}) : (A, I) \rightarrow (B, I)$. And, it also accepts a functor from $\mathbf{Optic}_{\mathcal{C}}$, which sends an optic $\langle l \mid r \rangle : (A, A') \rightarrow (B, B')$ to the learner with components

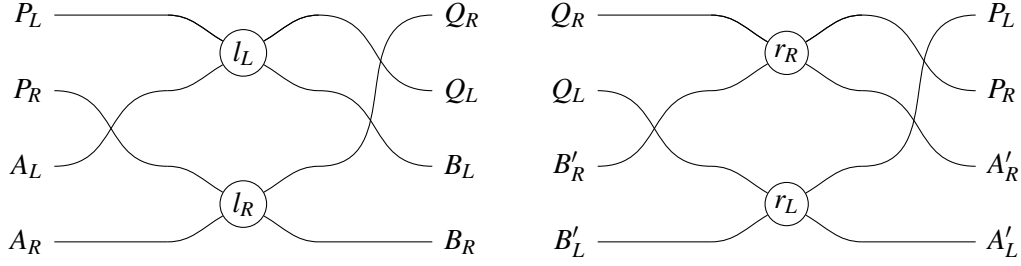
$$\begin{aligned} I \otimes A &\rightarrow A \xrightarrow{l} M \otimes B \\ M \otimes B' &\xrightarrow{r} A' \rightarrow I \otimes A' \end{aligned}$$

The observation that lenses include into learners as those with trivial parameter set was made in [7], and this fact is made especially clear by the characterisation $\mathbf{IntLearn}_{\mathcal{C}} \cong \mathbf{Para}(\mathbf{Lens}_{\mathcal{C}})$ given in [2].

Proposition 1.8. $\mathbf{Learn}_{\mathcal{C}}$ is symmetric monoidal, with action on objects given by $(A, A') \otimes (B, B') := (A \otimes B, B' \otimes A')$ and action on morphisms

$$(l_L \mid r_L) \otimes (l_R \mid r_R) : (A_L, A'_L) \otimes (A_R, A'_R) \rightarrow (B_L, B'_L) \otimes (B_R, B'_R)$$

given by



Proof. Similar to [15, Theorem 2.0.12], but using a version of the ‘switched’ tensor from [15, Definition 2.1.1]. The key facts to check are that the action on morphisms does not depend on the choice of representative and that $\otimes$ is functorial. The first is clear from the above diagrams: for example, a morphism on the Q_L string between l_L and r_L can be slid from l_L to r_L in the string diagram (using the coend relation to hop the gap), and similarly for the three strings corresponding to Q_R , P_L and P_R .

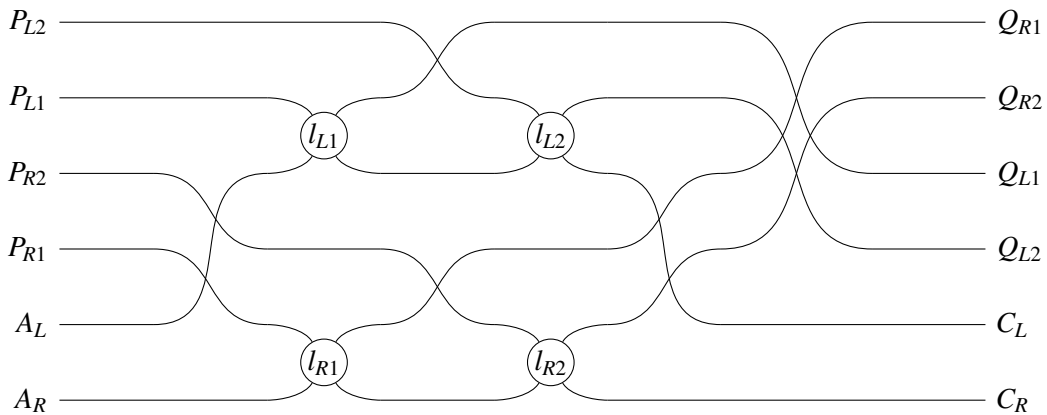
Functoriality of $\otimes$ follows from the equality of the following diagrams, depicting the left component of the morphisms

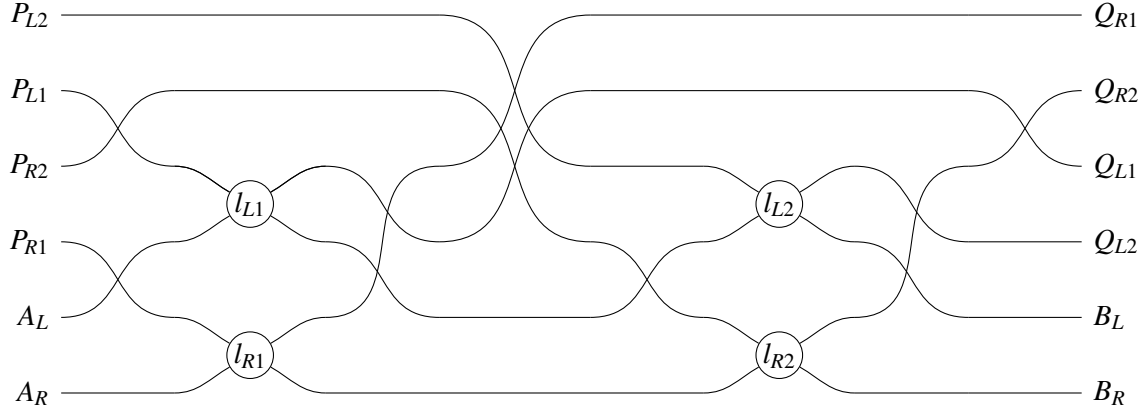
$$((l_{L2} \mid r_{L2}) \circ (l_{L1} \mid r_{L1})) \otimes ((l_{R2} \mid r_{R2}) \circ (l_{R1} \mid r_{R1}))$$

and

$$((l_{L2} \mid r_{L2}) \otimes (l_{R2} \mid r_{R2})) \circ ((l_{L1} \mid r_{L1}) \otimes (l_{R1} \mid r_{R1}))$$

respectively. (Imagine dragging the l_{L1} node below the P_{R2} string to the left, and the l_{L2} node below the Q_{R1} string to the right.)





The latter diagram is almost exactly the correct one, other than the outer twists on the pairs (P_{L1}, P_{R2}) and (Q_{R2}, Q_{L1}) . These are cancelled with similar twists in the right component using the coend relations.

As in the **Optic** setting, the structure morphisms (including the symmetry isomorphism) are constructed as the image of the structure morphisms of $\mathcal{C} \times \mathcal{C}^{\text{op}}$, and functoriality of the inclusion guarantees that the equations of a symmetric monoidal category hold. $\square$

2 Duality

For extensional learners, there is an obvious involution $(-)^* : \mathbf{Learn} \rightarrow \mathbf{Learn}^{\text{op}}$ that comes into view by virtue of the new symmetrical formulation: just switch the two components!

Proposition 2.1. *There is a (strict) involutive symmetric monoidal functor $(-)^* : \mathbf{Learn} \rightarrow \mathbf{Learn}^{\text{op}}$ given by $(A, A')^* := (A', A)$ and $(L \mid R)^* := (R \mid L)$.*

Proof. On objects, we have

$$((A, A') \otimes (B, B'))^* = (A \otimes B, B' \otimes A')^* = (B' \otimes A', A \otimes B) = (B', B) \otimes (A', A) = (B, B')^* \otimes (A, A')^*$$

Preservation of identity, composition and tensor of morphisms is clear by observing that their definitions are exactly symmetric between each component. $\square$

The resulting strictness of this functor is why it is convenient to switch the objects in the right component of $(A, A') \otimes (B, B') \equiv (A \otimes B, B' \otimes A')$.

Remark 2.2. Let us calculate how this duality acts on intensional learners. Stepping through the equivalence between intensional learners and their coend formulation, the dual of a learner $(P, I, U, r) : (A, A') \rightarrow (B, B')$ is a learner $(P^*, I^*, U^*, r^*) : (B', B) \rightarrow (A', A)$ with parameter set $P^* := P \times A$, and

$$\begin{aligned} I^* : (P \times A) \times B' &\rightarrow A' \\ I^*((p, p_a), b') &:= r(p, p_a, b') \\ U^* : (P \times A) \times B' \times A &\rightarrow (P \times A) \\ U^*((p, p_a), b', a) &:= (U(p, p_a, b'), a) \\ r^* : (P \times A) \times B' \times A &\rightarrow B \\ r^*((p, p_a), b', a) &:= I(U(p, p_a, b'), a) \end{aligned}$$

If we do this twice, we return to a learner $(A, A') \rightarrow (B, B')$ with parameter set $P^{**} := P \times A \times B'$, and

$$\begin{aligned} I^{**} &: (P \times A \times B') \times A && \rightarrow B \\ I^{**} &((p, p_a, p_{b'}), a) && := I(U(p, p_a, p_{b'}), a) \\ U^{**} &: (P \times A \times B') \times A \times B' && \rightarrow (P \times A \times B') \\ U^{**} &((p, p_a, p_{b'}), a, b') && := (U(p, p_a, p_{b'}), a, b') \\ r^{**} &: (P \times A \times B') \times A \times B' && \rightarrow A' \\ r^{**} &((p, p_a, p_{b'}), a, b') && := r(U(p, p_a, p_{b'}), a, b') \end{aligned}$$

Under intensional equivalence this is not equal to the original learner, but under extensional equivalence it is: use $U : P \times A \times B' \rightarrow P$ as the function f between parameter sets, and the identity $\text{id} : P \times A \times B' \rightarrow P \times A \times B'$ as the diagonal filler $\hat{U}$. This double-dual learner is extensionally identical to the original learner, but intensionally it is ‘running one training datum behind’: when fed a new element of $A \times B'$, it updates P using the pair remembered in the parameter set and stores the provided element of $A \times B'$ for next time.

An object (A, A') and its counterpart (A', A) are related by cup and cap morphisms:

Definition 2.3. For any object (A, A') , define the cup

$$\eta_{(A, A')} : (I, I) \rightarrow (A, A') \otimes (A, A')^* = (A \otimes A', A \otimes A')$$

as the learner with $P := A \otimes A'$ and $Q := I$ and the obvious maps

$$\begin{aligned} (A \otimes A') \otimes I &\rightarrow I \otimes (A \otimes A') \\ I \otimes (A \otimes A') &\rightarrow (A \otimes A') \otimes I \end{aligned}$$

and define the cap

$$\varepsilon_{(A, A')} : (A' \otimes A, A' \otimes A) = (A, A')^* \otimes (A, A') \rightarrow (I, I)$$

as the learner with $P := I$ and $Q := A' \otimes A$ and the obvious maps

$$\begin{aligned} I \otimes (A' \otimes A) &\rightarrow (A' \otimes A) \otimes I \\ (A' \otimes A) \otimes I &\rightarrow I \otimes (A' \otimes A). \end{aligned}$$

These maps are dual, in that:

$$(\eta_{(A, A')})^* = \varepsilon_{(A, A')^*} \quad (\varepsilon_{(A, A')})^* = \eta_{(A, A')^*}$$

All signs are pointing to $(A, A')^*$ being the actual monoidal dual of (A, A') . Tragically,

Proposition 2.4. $\text{Learn}_{\mathcal{C}}$ is typically not compact closed.

Proof. Consider an object of the form (A, I) . After cancelling some monoidal units, the composite learner

$$(A, I) \rightarrow (A, I) \otimes ((A, I)^* \otimes (A, I)) \rightarrow ((A, I) \otimes (A, I)^*) \otimes (A, I) \rightarrow (A, I)$$

corresponding to the snake equation for (A, I) has $P := A$ and $Q := A$, and components

$$\begin{aligned} A \otimes A &\xrightarrow{s_{A,A}} A \otimes A \\ A \otimes I &\xrightarrow{\text{id}} A \otimes I. \end{aligned}$$

This is typically not equal to the identity learner; a priori there are no non-trivial maps involving A whatsoever, as can be seen by setting $\mathcal{C}$ to be a commutative monoid considered as a discrete category.

As an intensional learner, the above learner is similar to the identity map but with its output “delayed one step”, and has $P := A$ with maps $I(p, a) := p$ and $U(p, b, a) := a$. $\square$

Learners do have the following weaker structure, much like the teleological categories discussed in [9, Section 5] and [15, Section 2.1] but with cups as well as caps.

Proposition 2.5. *The families of morphisms*

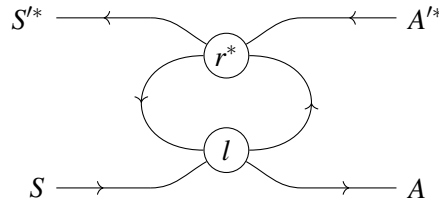
$$\begin{aligned} \eta_{(A,A')} : (I, I) &\rightarrow (A, A') \otimes (A, A')^* \\ \varepsilon_{(A,A')} : (A, A')^* \otimes (A, A') &\rightarrow (I, I) \end{aligned}$$

are monoidal, and are extranatural with respect to morphisms in the image of $\iota : \mathcal{C} \times \mathcal{C}^{\text{op}} \rightarrow \mathbf{Learn}_{\mathcal{C}}$. Furthermore, $s_{(A,A'), (A,A')^*} \eta_{(A,A')} = \eta_{(A,A')^*}$ and $\varepsilon_{(A,A')} s_{(A,A'), (A,A')^*} = \varepsilon_{(A,A')^*}$, where $s_{(A,A'), (A,A')^*}$ denotes the symmetry morphism.

Proof. The definitions of η and ε are so simple that these are all easily checked by direct calculation. $\square$

We have a canonical decomposition of any learner using these morphisms, in the same style as [15, Proposition 2.1.10].

Lemma 2.6. *Every learner $(l \mid r) : (S, S') \rightarrow (A, A')$ is equal to the learner described by the diagram*



recalling that we silently include objects and morphisms of $\mathcal{C}$ in $\mathbf{Learn}_{\mathcal{C}}$ using ι .

Proof. This can be checked by direct calculation, cancelling all the occurrences of the monoidal unit created by the use of ι . $\square$

The snake equations are precisely what is missing for $\mathbf{Learn}_{\mathcal{C}}$ to be compact closed.

Definition 2.7. The category $\mathbf{Atemp}_{\mathcal{C}}$ of *atemporal learners* is the quotient of $\mathbf{Learn}_{\mathcal{C}}$ obtained by identifying the snake diagram

$$(A, A') \rightarrow ((A, A') \otimes (A, A')^*) \otimes (A, A') \rightarrow (A, A') \otimes ((A, A')^* \otimes (A, A')) \rightarrow (A, A')$$

with the identity morphism for every object (A, A') .

Trivialising just one of the snake diagrams also trivialises the other: the dual snake diagram is the image of this one under the strictly monoidal functor $-^*$.

Proposition 2.8. $\mathbf{Atemp}_{\mathcal{C}}$ is compact closed.

Proof. The only missing property is extranaturality of the cup and cap with respect to *all* morphisms in $\mathbf{Atemp}_{\mathcal{C}}$. The snake equations holding implies that η and ε are extranatural with respect to each other:

and similarly with the roles reversed.

From Lemma 2.6, it follows that η and ε are natural with respect to *every* map in $\mathbf{Atemp}_{\mathcal{C}}$. Decomposing a map as via the Lemma, we can then see:

The unit is similar. □

3 Free Compact Closure

We conclude by showing that this $\mathbf{Atemp}_{\mathcal{C}}$ is the free compact closed category on a symmetric monoidal category $\mathcal{C}$.

There is an existing construction of the free compact closed category given by composing three simpler constructions: [13] constructs the free *feedback* monoidal category on a monoidal category before quotienting it to form the free *traced* monoidal category. Then [12] constructs the free compact closed category on a traced monoidal category. In the literature one also finds the free compact closed category on a bare category [14], and the free compact closed category on a *closed* monoidal category [3], but these are less comparable to the construction given here.

As a first step, any functor from $\mathcal{C}$ into a compact closed category factors through $\mathbf{Atemp}_{\mathcal{C}}$.

Theorem 3.1. *Suppose $\mathcal{C}$ is a symmetric monoidal category and $\mathcal{D}$ is a compact closed category. For any symmetric monoidal functor $F : \mathcal{C} \rightarrow \mathcal{D}$, there is a factorisation*

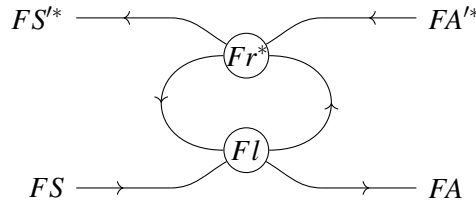
$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{F} & \mathcal{D} \\ & \searrow \iota \quad \nearrow \hat{F} & \\ & \mathbf{Atemp}_{\mathcal{C}} & \end{array}$$

Here we follow the shape of the argument in [12, Section 5].

Proof. Any object (A, A') of $\mathbf{Atemp}_{\mathcal{C}}$ is isomorphic to

$$(A, A') \cong (A', I)^* \otimes (A, I) = (\iota A')^* \otimes \iota A$$

which forces us to define $\hat{F}(A, A') := (FA')^* \otimes FA$. On a morphism $(l \mid r)$, define $\hat{F}(l \mid r)$ as indicated in the diagram

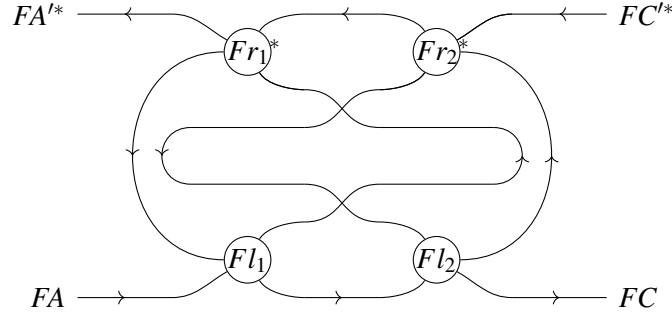


This is well defined with respect to the quotient on $\mathbf{Learn}_{\mathcal{C}}$, because the snake equations already hold in $\mathcal{D}$. The functor $\hat{F}$ behaves nicely with the inclusion $\iota : \mathcal{C} \times \mathcal{C}^{\text{op}} \rightarrow \mathbf{Atemp}_{\mathcal{C}}$, because $\hat{F}(\iota(f, g)) = g^* \otimes f$ for any two morphisms of $\mathcal{C}$.

There are now several things to check. First, that $\hat{F}$ is a functor. Preservation of identity follows because the cup and cap for the unit object in $\mathcal{D}$ are the unitors. Preservation of composition is the equivalence of the following diagrams:

$$\begin{array}{ccc} FA'^* & \xleftarrow{\quad} & \text{Diagram 1} & \xleftarrow{\quad} & FC'^* \\ & & \text{Diagram 1} & & \\ FA & \xrightarrow{\quad} & \text{Diagram 1} & \xrightarrow{\quad} & FC \end{array} = \begin{array}{ccc} FA'^* & \xleftarrow{\quad} & \text{Diagram 2} & \xleftarrow{\quad} & FC'^* \\ & & \text{Diagram 2} & & \\ FA & \xrightarrow{\quad} & \text{Diagram 2} & \xrightarrow{\quad} & FC \end{array}$$

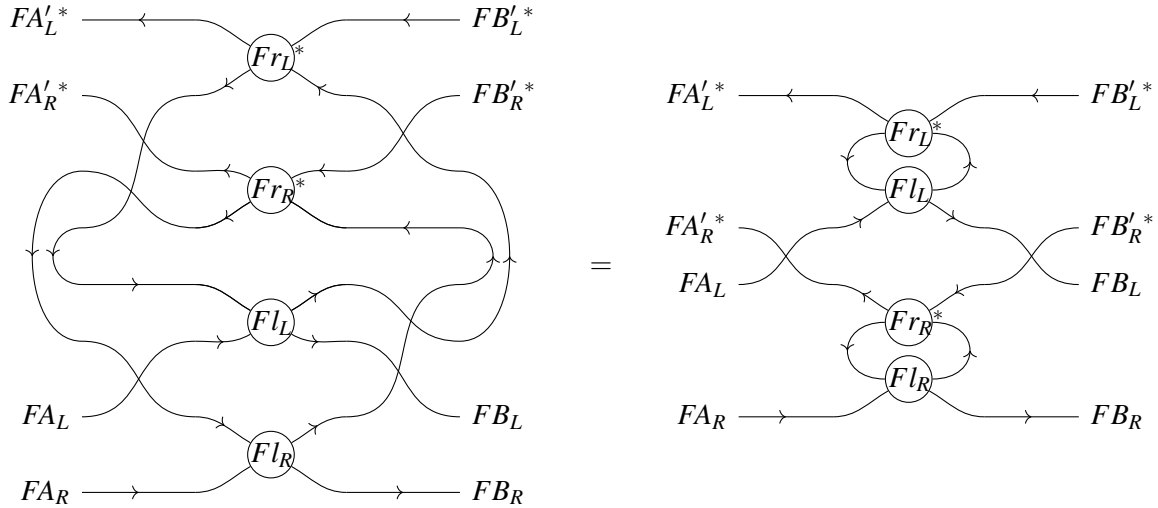
(The diagrams consist of two separate circular nodes, each with a top node labeled Fr and a bottom node labeled Fl, connected by curved arrows between them.)



We must equip $\hat{F}$ with the structure of a monoidal functor. For the structure isomorphism $\phi : \hat{F}(A_L, A'_L) \otimes \hat{F}(A_R, A'_R) \rightarrow \hat{F}(A_L \otimes A_R, A'_R \otimes A'_L)$, we take the composite

$$\begin{aligned}
 & \hat{F}(A_L, A'_L) \otimes \hat{F}(A_R, A'_R) \\
 &= (FA_L \otimes FA'_L) \otimes (FA_R \otimes FA'_R) \\
 &\rightarrow (FA_L \otimes FA_R) \otimes (FA'_L \otimes FA'_R) \\
 &\rightarrow (FA_L \otimes FA_R) \otimes (FA'_R \otimes FA'_L)^* \\
 &\rightarrow F(A_L \otimes FA_R) \otimes (F(A'_R \otimes A'_L))^* \\
 &= \hat{F}(A_L \otimes A_R, A'_R \otimes A'_L) \\
 &= \hat{F}((A_L, A'_L) \otimes (A_R, A'_R))
 \end{aligned}$$

That this gives a monoidal functor follows from the equivalence of the diagrams



The functor is also symmetric monoidal, as the symmetry morphisms of $\mathbf{Atemp}_c$ are inherited from the symmetry morphisms of $\mathcal{C} \times \mathcal{C}^{\text{op}}$ via ι . $\square$

Let $\mathbf{SymMon}_g$ denote the 2-category of symmetric monoidal categories, monoidal functors and monoidal natural *isomorphisms*, and let $\mathbf{Comp}$ denote the 2-category of compact closed categories, monoidal functors and monoidal natural transformations. Recall that no condition on monoidal functors between compact closed categories is necessary: duals are always preserved up to canonical isomorphism. We also do not need to explicitly restrict the 2-cells in $\mathbf{Comp}$ to be invertible, because *any* such monoidal natural transformation is invertible:

$$GA \rightarrow (G(A^*))^* \xrightarrow{(\alpha_{A^*})^*} (F(A^*))^* \rightarrow FA$$

Proof. Suppose $\mathcal{D}$ is compact closed. For two symmetric monoidal functors $F, G: \mathbf{Atemp}_{\mathcal{C}} \rightarrow \mathcal{D}$, the restriction map along $\iota_{\mathcal{C}}$ gives a map

We claim that this restriction map is a bijection.

$$\begin{aligned} F(A, A') &= F(\iota A \otimes (\iota A')^*) \rightarrow F\iota A \otimes F(\iota A'^*) \\ &\rightarrow F\iota A \otimes (F\iota A')^* \xrightarrow{\beta_A \otimes (\beta_{A'}^{-1})^*} G\iota A \otimes (G\iota A')^* \\ &\rightarrow G\iota A \otimes G(\iota A')^* \rightarrow G(\iota A \otimes \iota A'^*) = G(A, A') \end{aligned}$$
$$\begin{array}{ccccc} FA'^* & \leftarrow & \beta_{A'}^{-1} & \leftarrow & GA'^* \\ FA & \rightarrow & \beta_A & \rightarrow & GA \end{array}$$

The figure consists of two circuit diagrams separated by an equals sign, indicating they are equivalent. Each diagram has four input/output nodes on the left and right, and two central nodes.

Left Diagram:

- Left Nodes (top to bottom):** FA'^* , β_P^{-1*} , β_P , FA .
- Right Nodes (top to bottom):** GB'^* , GB .
- Central Nodes:** Gr^* (top) and Gl (bottom).
- Connections:**
 - FA'^* and β_P^{-1*} are connected to Gr^* .
 - Gr^* is connected to GB'^* .
 - β_P and FA are connected to Gl .
 - Gl is connected to GB .
 - There are curved double-headed arrows between β_P^{-1*} and β_P , and between Gr^* and Gl .

Right Diagram:

- Left Nodes (top to bottom):** FA'^* , FA .
- Right Nodes (top to bottom):** GB'^* , GB .
- Central Nodes:** Fr^* (top) and Fl (bottom).
- Connections:**
 - FA'^* and Fr^* are connected.
 - Fr^* is connected to GB'^* .
 - FA and Fl are connected.
 - Fl is connected to GB .
 - There are curved double-headed arrows between Fr^* and Fl , and between GB'^* and GB .

Finally, we must show that this process is inverse to restriction along ι . It is clear from the definition that $\overline{\beta}\iota = \beta$. The interesting case is the converse, that $\overline{\alpha}\iota = \alpha$ for a monoidal isomorphism $\alpha : F \rightarrow G$.

We verify

$$\begin{aligned}
F(A, A') &\xrightarrow{\overline{\alpha}\iota_{(A, A')}} G(A, A') \\
&= F(A, A') \xrightarrow{\sim} F\iota A \otimes (F\iota A')^* \xrightarrow{\alpha_{\iota A} \otimes (\alpha_{\iota A'}^{-1})^*} G\iota A \otimes (G\iota A')^* \xrightarrow{\sim} G(A, A') \\
&= F(A, A') \xrightarrow{\sim} F\iota A \otimes (F\iota A')^* \xrightarrow{\alpha_{\iota A} \otimes (\alpha_{\iota A'}^{-1})^*} G\iota A \otimes (G\iota A')^* \xrightarrow{\sim} G(A, A') \\
&= F(A, A') \xrightarrow{\sim} F\iota A \otimes (F\iota A')^* \xrightarrow{\sim} F\iota A \otimes (F(\iota A')^*)^{**} \xrightarrow{\alpha_{\iota A} \otimes (\alpha_{(\iota A')^*})^{**}} G\iota A \otimes (G(\iota A')^*)^{**} \\
&\quad \xrightarrow{\sim} G\iota A \otimes (G\iota A')^* \xrightarrow{\sim} G(A, A') \\
&= F(A, A') \xrightarrow{\sim} F\iota A \otimes (F(\iota A')^*) \xrightarrow{\alpha_{\iota A} \otimes (\alpha_{(\iota A')^*})} G\iota A \otimes (G\iota A')^* \xrightarrow{\sim} G(A, A') \\
&= F(A, A') \xrightarrow{\alpha_{(A, A')}} G(A, A')
\end{aligned}$$

by expanding the explicit description of the inverse of $\alpha_{\iota A'}$ given by Proposition 3.2, and using the monoidalness of α at the last step.

From this we conclude that restriction along the inclusion ι is an equivalence of categories between $\mathbf{Comp}(\mathbf{Atemp}_{\mathcal{C}}, \mathcal{D})$ and $\mathbf{SymMon}_g(\mathcal{C}, \mathcal{D})$, making ι a *biuniversal arrow*. It follows abstractly from [5, Theorem 9.17] that **Atemp** assembles into a bifunctor $\mathbf{Atemp} : \mathbf{SymMon}_g \rightarrow \mathbf{Comp}$, left biadjoint to the inclusion. $\square$

Remark 3.4. The restriction to $\mathbf{SymMon}_g$ with natural *isomorphisms* as 2-cells is necessary. As pointed out by [8], this assumption is erroneously missing from the similar theorem given in [12].

References

- [1] Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore & Mario Román (2024): *Profunctor optics: a categorical update*. *Compositionality* 6(1), pp. 1–39, doi:[10.32408/compositionality-6-1](https://doi.org/10.32408/compositionality-6-1).
- [2] Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2022): *Categorical Foundations of Gradient-Based Learning*. In Ilya Sergey, editor: *Programming Languages and Systems*, Springer International Publishing, Cham, pp. 1–28, doi:[10.1007/978-3-030-99336-8_1](https://doi.org/10.1007/978-3-030-99336-8_1).
- [3] B. J. Day (1977): *Note on compact closed categories*. *J. Austral. Math. Soc. Ser. A* 24(3), pp. 309–311, doi:[10.1017/s1446788700020334](https://doi.org/10.1017/s1446788700020334).
- [4] Emilio Faro (2008): *On the trace of an endofunctor of a small category*. Technical Report. Available at <https://ddd.uab.cat/record/44451>.
- [5] Thomas M. Fiore (2006): *Pseudo limits, biadjoints, and pseudo algebras: categorical foundations of conformal field theory*. *Mem. Amer. Math. Soc.* 182(860), pp. x+171, doi:[10.1090/memo/0860](https://doi.org/10.1090/memo/0860).
- [6] Brendan Fong & Michael Johnson (2019): *Lenses and Learners*. In: *Proceedings of the 8th International Workshop on Bidirectional Transformations (Bx 2019)*, pp. 16–29. Available at <https://ceur-ws.org/Vol-2355/paper2.pdf>.
- [7] Brendan Fong, David Spivak & Rémy Tuyéras (2019): *Backprop as functor: a compositional perspective on supervised learning*, doi:[10.1109/LICS.2019.8785665](https://doi.org/10.1109/LICS.2019.8785665).
- [8] Masahito Hasegawa & Shin-ya Katsumata (2010): *A note on the biadjunction between 2-categories of traced monoidal categories and tortile monoidal categories*. *Math. Proc. Cambridge Philos. Soc.* 148(1), pp. 107–109, doi:[10.1017/S0305004109002606](https://doi.org/10.1017/S0305004109002606).

- [9] Jules Hedges (2017): *Coherence for lenses and open games*. arXiv:[1704.02230](#).
- [10] James Hefford & Cole Comfort (2023): *Coend Optics for Quantum Combs*. In: *Proceedings of the Fifth International Conference on Applied Category Theory (ACT2022)*, *Electron. Proc. Theor. Comput. Sci. (EPTCS)* 380, EPTCS, pp. 63–76, doi:[10.4204/EPTCS.380.4](#).
- [11] André Joyal & Ross Street (1993): *Braided tensor categories*. *Adv. Math.* 102(1), pp. 20–78, doi:[10.1006/aima.1993.1055](#).
- [12] André Joyal, Ross Street & Dominic Verity (1996): *Traced monoidal categories*. *Math. Proc. Cambridge Philos. Soc.* 119(3), pp. 447–468, doi:[10.1017/S0305004100074338](#).
- [13] P. Katis, Nicoletta Sabadini & Robert F. C. Walters (2002): *Feedback, trace and fixed-point semantics*. 36, pp. 181–194, doi:[10.1051/ita:2002009](#). Fixed points in computer science (Florence, 2001).
- [14] G. M. Kelly & M. L. Laplaza (1980): *Coherence for compact closed categories*. *J. Pure Appl. Algebra* 19, pp. 193–213, doi:[10.1016/0022-4049\(80\)90101-2](#).
- [15] Mitchell Riley (2018): *Categories of Optics*. arXiv:[1809.00738](#).
- [16] Mitchell Riley & The CT Zulip Community (2021): *An involution on ‘Learn’*. Available at <https://categorytheory.zulipchat.com/#narrow/stream/229156-theory.3A-applied-category-theory/topic/an.20involution.20on.20'Learn'>.

Combs, Causality and Contractions in Atomic Markov Categories

Dario Stein Márk Széles

Radboud University Nijmegen, The Netherlands

{dario.stein,mark.szeles}@ru.nl

We present a counterexample showing that Markov categories with conditionals (such as **BorelStoch**) need not validate a natural scheme of axioms which we call contraction identities. These identities hold in every traced monoidal category, so in particular this shows that **BorelStoch** cannot be embedded in any traced monoidal category. We remedy this under the additional assumption of atomicity: Atomic Markov categories validate all contraction identities, and furthermore admit a notion of trace defined for non-signalling morphisms. We conclude that atomic Markov categories allow for an intrinsic theory of combs without having to assume an embedding into a compact-closed category.

1 Introduction

Markov categories with conditionals have emerged as a general and powerful framework for studying stochastic processes and notions such as conditioning and independence in an abstract way that generalizes reasoning in graphical models (e.g. [6, 13, 10, 9, 11, 17, 25]). An important challenge is to understand the *implicational theory* of such categories: assuming some equation or factorization is valid, which further equations can be derived from this? For example, such quasi-identities lie at the heart of *identifiability* in causal inference; see Figure 4 for a treatment of Pearl’s front-door adjustment in such categorical terms.

In this work, we are interested in a particular schema of implications which we call *contraction identities*¹. The simplest instance is the following implication for all morphisms $f_1, f_2 : X \otimes X \rightarrow Y$

$$\begin{array}{c} Y \quad X \\ \boxed{f_1} \\ \downarrow \quad \downarrow \\ X \quad V \end{array} = \begin{array}{c} Y \quad X \\ \boxed{f_2} \\ \downarrow \quad \downarrow \\ X \quad V \end{array} \Rightarrow \begin{array}{c} Y \quad Y \\ \boxed{f_1} \quad \boxed{f_2} \\ \downarrow \quad \downarrow \\ V \quad V \end{array} \quad (1)$$

Every contraction identity is formed by connecting an outgoing wire (here X) to an incoming one and pulling the strings tight. This must be done in an acyclic way to make sure the resulting diagram is meaningful – we need to avoid genuine feedback loops. We will give the full combinatorial definition of the contraction scheme in (30) using the language of free Markov categories [12] and hypergraphs.

We show using a simple counterexample that (1) does not hold in the category **BorelStoch**, despite it having conditionals. We remedy this under a common additional assumption called *atomicity*. Common categories such as **FinStoch**, **Gauss** and **SetMulti** are atomic, while **Stoch**, **BorelStoch** and categories modelling fresh name generation are not. Our main theorem 32 states that atomic Markov categories

¹we will shorten quasi-identities to identities from now on

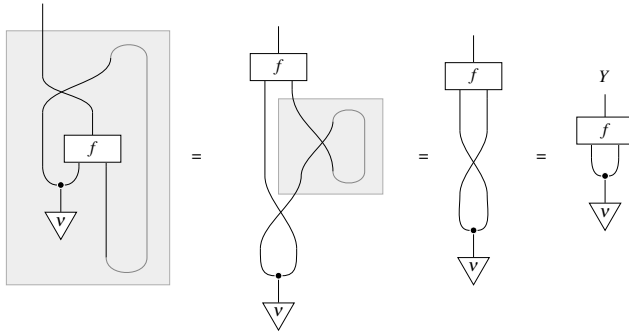
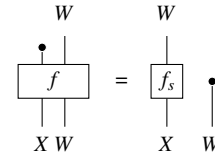


Figure 1: Derivation of (1) using traces

Figure 2: A morphism f satisfies the non-signalling condition if there exists an f_s such that the equation above holds.

with conditionals satisfy all contraction identities. As we will discuss next, this enables us to apply the convenient notions of causal traces and combs in such categories.

Causal Traces A *trace* on a monoidal category $\mathbf{C}$ is an operator which assigns to every morphism $f : X \otimes W \rightarrow Y \otimes W$ a morphism $\text{tr}_{X,Y}^W(f) : X \rightarrow Y$ which represents evaluating a feedback loop of W into itself. We depict the trace operation as in Figure 3.

The trace satisfies a number of axioms which encode the desired properties of such a feedback operation (see Section 3), and the graphical calculus reflects these [24]. We show that all contraction identities can be proved from the trace axioms (Proposition 31). For example in (1), we can obtain the consequent equation via the trace on Figure 1.

As a consequence of our counterexample, **BorelStoch** cannot be embedded in *any* traced monoidal category. We then establish the following converse result, in trying to obtain a canonical notion of trace on an atomic Markov category. It is hopeless to attempt this for an arbitrary morphism $f : X \otimes W \rightarrow Y \otimes W$. It is possible, however, if f satisfies the non-signalling condition, see Figure 2.

This means the input W is not needed to compute the output W . So the feedback loop created by the trace remains causal. Our central Definition 4 states that every atomic Markov category with conditionals admits a canonical notion of causal trace, that is a trace defined on non-signalling morphisms. We show that this construction satisfies a restricted version of the trace axioms, and compare with the notion of a partially traced category [14, 2] and traced ideal [1]. The properties of the causal trace suffice to derive all contraction identities (Propositions 31 and 32).

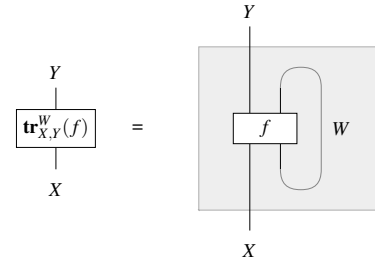


Figure 3: Graphical depiction of the trace operator

Combs and Causal Inference Combs are a useful tool for studying decompositions of string diagrams into more flexible shapes, with applications in quantum theory and causality [3, 20, 15, 22, 23, 18]. For example, Figure 4 expresses Pearl's famous *front-door adjustment* [21, §3.3.2] using combs and string diagram surgery.

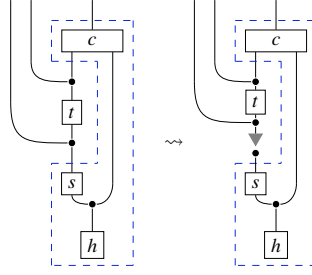


Figure 4: Determining the causal effect of s on c : Pre-intervention (left) and post-intervention distribution (right). Identifiability means any factorization on the left gives the same post-intervention distribution. We refer to [18] for the details of this example.

Informally, a comb C of type $(A, A') \rightarrow (B, B')$ is a diagram of type $A \rightarrow A'$ which features a *hole* of type $B \rightarrow B'$. This hole can be filled by appropriate morphisms h , leading to a composite $C[h]$. This way, a comb describes a second-order process and lends itself to be rendered in an evocative shape (see Figure 5a).

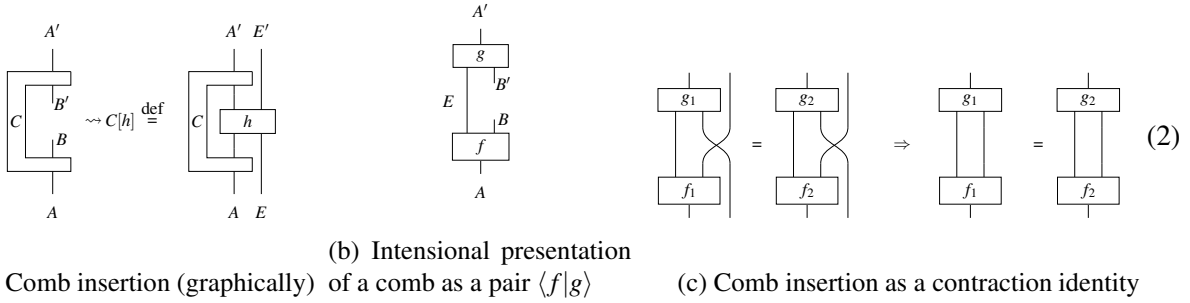


Figure 5: Combs and comb insertion

There are various inequivalent ways of formally defining what a comb is. An extensional definition is to define a comb as a morphism $A \otimes B' \rightarrow A' \otimes B$ in $\mathbf{C}$ which is non-signalling from B' to B . An intensional definition is as a pair of morphisms $\langle f|g \rangle$ with $f : A \rightarrow E \otimes B$ and $g : E \otimes B' \rightarrow A'$, under some kind of equivalence relation. We review these notions in Section 7 following [15].

Relating the different definitions of combs to each other is tricky: To go from the extensional definition to comb insertion is an instance of a contraction identity (Figure 5c).

For this reason, the theory of combs has commonly been developed in the setting of compact closed categories. There, second-order processes can be reduced to first-order processes by bending wires, and the various notions of comb are equivalent. The downside is that even when analyzing diagrams in a Markov category $\mathbf{C}$ using combs, we must assume that it comes with an embedding into a compact closed category (for example $\mathbf{FinStoch} \hookrightarrow \mathbf{Mat}(\mathbb{R}^+)$). Not only is this difficult for practical categories (such an embedding category requires developing a theory of exact conditioning, e.g. [5, 26]), but our counterexample shows that this is generally impossible in the absence of the atomicity axiom: Identity (2) is invalid in **BorelStoch**. We argue that causal traces are sufficient to develop an intrinsic theory of combs. We prove that in every atomic Markov category with conditionals, the extensional and intensional definitions of combs are equivalent.

Contributions This work is a shared refinement of three different developments [16, 15, 8]. The major starting point is [16], where the author develops a comprehensive theory of causality and contraction. Their key statement is Lemma 4.2.5, which they prove under the assumption of *universal dilations*. By focusing on Markov categories, we obtain a new entry point into their developments, as we prove Lemma 4.2.5 in our Lemma 12 under a different set of assumptions. Using that lemma as a starting point, we rederive a streamlined and self-contained presentation of their constructions which focuses more explicitly on connections with the literature on traced categories.

We then connect our theory to the different notions of combs compared in [15]. We generalize those results significantly from the compact-closed case to a large family of Markov categories, which removes the need to assume an embedding into a compact-closed category as done in [18]. We briefly return to the relationship between universal dilations and combs in Proposition 38. Our main contributions are the following.

1. We recognize the role of the atomicity axiom, which has been introduced in [8]. Our theorems are proven under the assumption of conditionals and atomicity, which are weaker than universal dilations.
2. Our counterexample (7) gives a precise reason for the failure of universality in **BorelStoch**.
3. The combinatorics of the contraction identities are elegantly phrased using free Markov categories.

Acknowledgements We'd like to thank many people for fruitful discussions, in particular Dylan Braithwaite, Tobias Fritz, Tomáš Gonda, Nicholas Gauguin Houghton-Larsen, and Nathaniel Virgo.

2 Markov Categories

This introductory material is taken from [6]. A *gs-monoidal category* or *CD-category* (due to [4]) is a symmetric monoidal category $(\mathbf{C}, \otimes, I)$ where every object is coherently equipped with the structure of a commutative comonoid $\Delta_X : X \rightarrow X \otimes X$, $\text{del}_X : X \rightarrow I$. We render these graphically as on Figure 6.

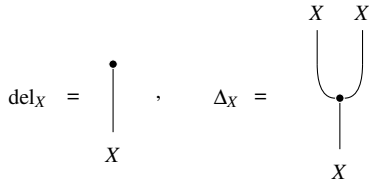


Figure 6: Graphical depiction of copy and delete maps

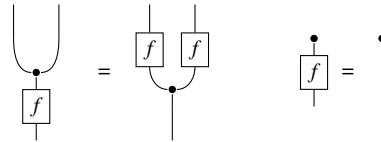


Figure 7: A map f is *copyable* (resp. *discardable*) if the equation on the left (resp. right) holds)

A *Markov category* is a CD category in which every morphism is discardable. Equivalently, this means del is a natural transformation, and therefore I is a final object. Copying is not assumed to be natural. From now on, let $\mathbf{C}$ be a Markov category.

We say that $\mathbf{C}$ *has conditionals* if for every $f : A \rightarrow X \otimes Y$ there exists $f|_X : X \otimes A \rightarrow Y$ such that the left equation of Figure 8 holds.

For morphisms $p : A \rightarrow X$ and $f : X \rightarrow Y$, a *Bayesian inverse* is a morphism $f_p^\dagger : A \otimes Y \rightarrow X$ such that the right equation on Figure 8 holds.

Conditionals and Bayesian inverses are mutually interdefinable, and we will be using both in the present article.

2.1 Example Categories

We briefly recall the Markov categories of interest, mainly to establish notation. For the full definitions, we ask the reader to consult the references.

Example 1. *The Markov category **FinStoch** consists of*

1. *objects are finite sets X*
2. *morphisms $p : X \rightarrow Y$ are stochastic matrices, with entries written $p(y|x) \in [0, 1]$, subject to the axiom*

$$\forall x \in X, \sum_{y \in Y} p(y|x) = 1$$

3. *composition is matrix multiplication, a.k.a. the Kolmogorov-Chapman equation*

$$(gf)(z|x) = \sum_y g(z|y)f(y|x)$$

The compact-closed category $\mathbf{Mat}(\mathbb{R}^+)$ is defined like **FinStoch** but allows arbitrary matrices with non-negative entries as morphisms. **FinStoch** corresponds to the subcategory of $\mathbf{Mat}(\mathbb{R}^+)$ of morphisms which are discardable.

Example 2. *The Markov category **BorelStoch** consists of*

1. *morphisms are standard Borel spaces (X, Σ_X)*
2. *morphisms $p : X \rightarrow Y$ are Markov kernels $p : X \times \Sigma_Y \rightarrow [0, 1]$. We write $p(x)$ for the measure $p(x, -) : \Sigma_Y \rightarrow [0, 1]$ on Y .*
3. *composition is Lebesgue integration*

$$(gf)(x, E) = \int g(y, E) f(x, dy)$$

For $x \in X$, we write δ_x for the Dirac distribution centered at x . We can consider every measurable function $f : (X, \Sigma_X) \rightarrow (Y, \Sigma_Y)$ as a **BorelStoch** morphism δ_f defined by $\delta_f(x) = \delta_{f(x)}$. As a slight abuse of notation, we will often write f for both the function and its induced Markov kernel.

Markov categories capture a variety of flavors of probability and nondeterminism. The following categories will only be mentioned in examples, so we give the references: **SetMulti** (sets and left-total relations) [6], **TychStoch** (Tychonoff spaces and continuous Markov kernels) [8], **Gauss** (Markov kernels built from multivariate normal distributions and linear maps) [6].

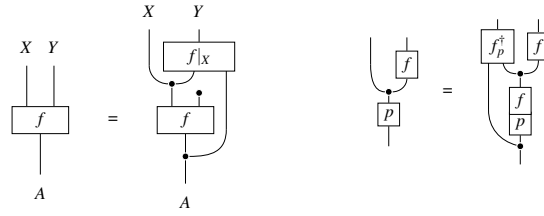


Figure 8: Conditionals and Bayesian inverses

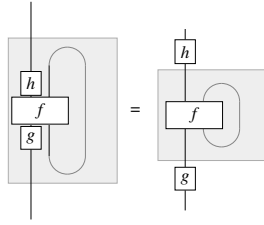
3 Traced Monoidal Categories

We briefly recall the notion of a traced monoidal category [19].

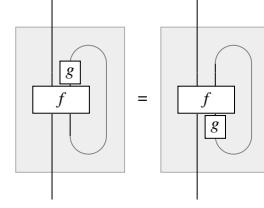
A traced monoidal category is a symmetric monoidal category $\mathbf{C}$ together with a family of operators

$$\mathbf{tr}_{X,Y}^W : \mathbf{C}(X \otimes W, Y \otimes W) \rightarrow \mathbf{C}(X, Y)$$

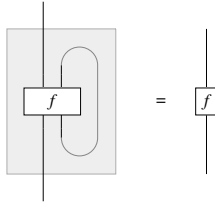
satisfying the axioms depicted in Figure 9. We will omit the grey shading of the trace boxes when it is clear from context how to interpret the trace.



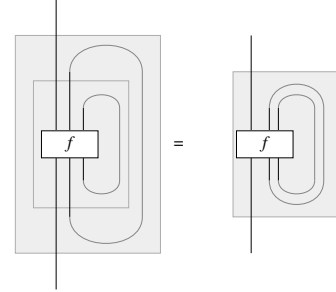
(a) Tightening (naturality in X, Y): For all $f : X \otimes W \rightarrow Y \otimes W$, $g : X' \rightarrow X$, $h : Y \rightarrow Y'$, we have $\mathbf{tr}^W((h \otimes \text{id}_W) \circ f \circ (g \otimes W)) = h \circ \mathbf{tr}^W(f) \circ g$



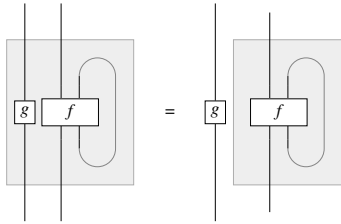
(b) Sliding (dinaturality in W): For all $f : X \otimes W \rightarrow Y \otimes W'$ and $g : W' \rightarrow W$, we have $\mathbf{tr}^W((\text{id}_Y \otimes g) \circ f) = \mathbf{tr}^{W'}(f \circ (\text{id}_X \otimes g))$



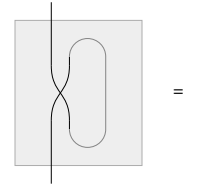
(c) Vanishing (coherence with I): For all $f : X \otimes I \rightarrow Y \otimes I$, we have $\mathbf{tr}^I(f) = (X \xrightarrow{\rho_X^{-1}} X \otimes I \xrightarrow{f} Y \otimes I \xrightarrow{\rho_Y} Y)$



(d) Associativity (coherence with $\otimes$): For all $f : X \otimes U \otimes V \rightarrow Y \otimes U \otimes V$ we have $\mathbf{tr}^{U \otimes V}(f) = \mathbf{tr}^U(\mathbf{tr}^V(f))$



(e) Superposition (strength): For all $f : X \otimes W \rightarrow Y \otimes W$ and $g : X' \rightarrow Y'$, we have $\mathbf{tr}^W(g \otimes f) = g \otimes \mathbf{tr}^W(f)$



(f) Yanking: $\mathbf{tr}^W(\text{swap}_{W,W}) = \text{id}_W$

Figure 9: Axioms of trace

4 Atomic Markov Categories

We recall the notion of atomicity in Markov categories, as well as the prerequisite notions of almost sure equality and absolute continuity found in [8]. Proofs of our propositions are to be found in the appendix. Let $\mathbf{C}$ be a Markov category.

Definition 3 (Almost sure equality – [8]). *Given a morphism $p : A \rightarrow X$, we say $f_1, f_2 : W \otimes X \rightarrow Y$ are p -almost surely equal (written $f_1 =_p f_2$) if the equation in Figure 10 holds.*

Definition 4 (Absolute continuity – [8]). *For two morphisms $p : A \rightarrow X$ and $q : B \rightarrow X$, we say that p is absolutely continuous with respect to q (written $p \ll q$), if $f_1 =_q f_2$ implies $f_1 =_p f_2$ for all $f_1, f_2 : W \otimes X \rightarrow Y$.*

We note the quantification over arbitrary W which is stronger than earlier definitions (e.g. [10]). These abstract notions capture the usual definitions of almost sure equality in our example categories (see [8, Section 2.2]):

1. in **FinStoch**, f_1, f_2 are p -almost surely equal if $f_1(y|x) = f_2(y|x)$ for all x with $p(x|a) > 0$ for some $a \in A$. We have $p \ll q$ if $\text{supp}(p) \subseteq \text{supp}(q)$ where

$$\text{supp}(p) = \{x \in X : \exists a \in A, p(x|a) > 0\}$$

2. in **BorelStoch**, f_1, f_2 are p -almost surely equal if the set $D = \{x : f_1(x) \neq f_2(x)\}$ satisfies $p(D|a) = 0$ for all $a \in A$. We have $p \ll q$ if for every every measurable subset $S \subseteq X$, we have

$$(\forall b \in B, q(S|b) = 0) \Rightarrow \forall a \in A, p(S|a) = 0$$

Definition 5 (Atomicity – [8]). *We call a morphism $p : A \rightarrow X$ atomic if $\Delta_X \circ p \ll p \otimes p$. We call the Markov category $\mathbf{C}$ atomic if every morphism in it is atomic.*

The following characterizations are known from [8], but we repeat them because they are instructive.

Example 6. *For every morphism $p : A \rightarrow X$ in **FinStoch**, we have*

$$(p \otimes p)(x_1, x_2 | a_1, a_2) = p(x_1 | a_1) p(x_2 | a_2), \quad (\Delta_X \circ p)(x_1, x_2 | a) = \begin{cases} p(x_1 | a), & x_1 = x_2 \\ 0, & \text{otherwise} \end{cases}$$

so it holds that $\text{supp}(\Delta_X \circ p) \subseteq \text{supp}(p \otimes p)$. Therefore **FinStoch** is atomic.

The name *atomicity* suggests that this property fails for distributions which are atomless, such as the Lebesgue measure. For a morphism $p : A \rightarrow X$ in **BorelStoch**, we define its set of atoms $\mathcal{A} \subseteq X$ as

$$\mathcal{A} = \{x \in X : \exists a \in A, p(\{x\}|a) > 0\}$$

We call p *completely atomic* if $p(\mathcal{A}|a) = 1$ for all $a \in A$, i.e. its probability mass is fully concentrated on its atoms. It is shown in [8, Theorem 3.2.7] that p is atomic in the sense of Definition 5 if and only if it is completely atomic. It is easy to see that the Lebesgue measure ν on the interval $[0, 1]$ is *not* atomic, as its set of atoms $\mathcal{A}$ is empty. We will elaborate a concrete counterexample, showing that $\Delta \circ \nu \not\ll \nu \otimes \nu$.

Example 7. **BorelStoch** is not atomic.

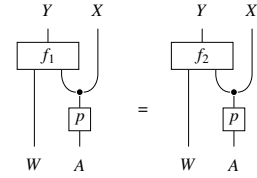


Figure 10: Almost sure equality

Proof. Let $X = [0, 1]$ and $\nu : I \rightarrow X$ be the Lebesgue measure. We will construct two measurable functions which are almost-sure equal with respect to $\nu \otimes \nu$ (the Lebesgue measure on the square), but not with respect to $\Delta \circ \nu$ (the pushforward measure on the diagonal).

We take these functions to be the characteristic functions of the empty set and the diagonal respectively, i.e. $e, f : X \otimes X \rightarrow \{0, 1\}$ given by $e(x, y) = [x = y]$, $f(x, y) = 0$. Then $e =_{\nu \otimes \nu} f$, because the set $\{(x, y) : e(x, y) \neq f(x, y)\}$ has measure 0 under $\nu \otimes \nu$. The following equality holds in string diagrams

However $e \neq_{\Delta \circ \nu} f$, because the set $\{(x, y) : e(x, y) \neq f(x, y)\}$ has measure 1 under $\Delta \circ \nu$! Similarly, the following string diagrams are *not* equal.

This shows $\Delta \circ \nu \not\leq \nu \otimes \nu$. □

As a simple corollary, we obtain the following:

Example 8. **BorelStoch** does not satisfy the contraction identity (1) from the introduction.

Proof.

□

This type of counterexample can easily be adapted to other Markov categories which feature atomless distributions, such as quasi-Borel spaces or nominal sets ([25]). As we will see, such counterexamples are impossible when the category in question is atomic.

A subtle caveat is that the notion of atomicity is dependent on the surrounding Markov category: The standard normal distribution $\mathcal{N}(0, 1)$ is non-atomic in **BorelStoch**, but it *is* atomic in the subcategory **Gauss**. More generally, the category **TychStoch** of Tychonoff spaces and continuous Markov kernels is atomic [8], despite featuring atomless measures: Note that the map e from the counterexample 8 fails to be continuous.

We return to the abstract properties of atomic morphisms.

Proposition 9. *In a Markov category $\mathbf{C}$, the following morphisms p are always atomic*

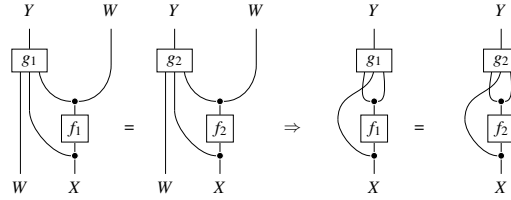
1. *morphisms of ‘full support’, in the sense that $f_1 =_p f_2$ implies $f_1 = f_2$.*
2. *deterministic morphisms*
3. *if $\mathbf{C}$ satisfies the causality axiom ([6, 11.31]), g is deterministic, and p is atomic, then the composite $g \circ p$ is atomic*

Note that atomic morphisms are generally not closed under composition, and atomicity cannot be checked on points (see Example 39). We now show that the atomicity axiom plus conditionals imply a basic contraction identity. These will be sufficient to prove all of them (5.1). We need the following technical notion to rule out pathological cases:

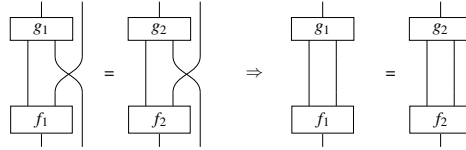
Definition 10. An object W of a Markov category is called *cancellable* if $\text{del}_W \otimes f_1 = \text{del}_W \otimes f_2$ implies $f_1 = f_2$. We call the Markov category *cancellative* if every object W is cancellable.

This condition is called *normality* in [16]. In practical examples, most objects are cancellable, for example any object that admits a state $I \rightarrow W$. In **FinStoch** and **BorelStoch**, every object except $W = \emptyset$ is cancellable. We may formally restrict our attention to sub-Markov categories **FinStoch**^{*}, **BorelStoch**^{*} on non-empty objects, or simply assume cancellability when needed.

Lemma 11. In an atomic Markov category, we have for all $f_1, f_2 : X \rightarrow W$ and $g_1, g_2 : W \otimes X \otimes W \rightarrow Y$ that



Lemma 12. Let $\mathbf{C}$ be an atomic Markov category with conditionals. Then



Note that this is precisely the statement of Lemma 4.2.5 of [16] in our setting.

5 Causal Traces

Definition 13. We say a morphism $f : X \otimes W' \rightarrow Y \otimes W$ is *non-signalling* (from W' to W) if there exists a morphism $f_s : X \rightarrow W$ such that

$$\begin{array}{c} W \\ \bullet \\ \boxed{f} \\ \text{---} \\ X \quad W' \end{array} = \begin{array}{c} W \\ \boxed{f_s} \\ \text{---} \\ X \quad W' \end{array}$$

This captures the intuition that in order to determine W , we don't need access to W' (but we do to compute the joint output Y).

For non-signalling morphisms, we can hope to define a canonical trace as follows.

Proposition 14. Let $\mathbf{C}$ be a Markov category with conditionals. Then $f : X \otimes W' \rightarrow Y \otimes W$ is non-signalling if and only if it can be written in the following form

$$\begin{array}{c} Y \quad W \\ \boxed{f} \\ \text{---} \\ X \quad W \end{array} = \begin{array}{c} Y \quad W \\ \boxed{f_p} \\ \text{---} \\ \boxed{f_s} \\ \text{---} \\ X \quad W \end{array} \quad (3)$$

In this article, we will call the form (3) a disintegration of f . Note that while f_s is unique (assuming cancellability), f_p is not.

Definition 15. Let $\mathbf{C}$ be an atomic Markov category with conditionals, and $f : X \otimes W \rightarrow Y \otimes W$ be non-signalling. Then we define the causal trace $\mathbf{tr}^W(f) : X \rightarrow Y$ in terms of any disintegration as

$$\text{Diagram (4)} \quad (4)$$

By Lemma 11, this definition does not depend on the choice of disintegration.

The definition of the causal trace is unique in that it is forced via the trace axiom. For a non-signalling morphism f , any trace must send f to (4). The special role of non-signalling is also reflected in the following proposition.

Proposition 16. Let $\mathbf{C}$ be a traced semicartesian monoidal category. If $f : X \otimes W \rightarrow Y \otimes W$ is a discardable morphism, then $\mathbf{tr}^W(f) : X \rightarrow Y$ need not be discardable, unless f is non-signalling.

We spell out concretely how to compute the trace in **FinStoch**:

Example 17. Let $f : X \otimes W \rightarrow Y \otimes W$ be a non-signalling morphism in **FinStoch**, and assume $W \neq \emptyset$. As we can compute the causal trace using any embedding into a traced category, such as the usual embedding into $\mathbf{Mat}(\mathbb{R})$. There we have

$$\mathbf{tr}^W(f)(y|x) = \sum_w f(y, w|x, w)$$

This will not define a normalized probability kernel for a general f . However, if f is non-signalling then the sum $\sum_y f(y, w|x, w')$ does not depend on w' . Hence by fixing any $w_0 \in W$, we can prove normalization

$$\sum_y \mathbf{tr}^W f(y|x) = \sum_y \sum_w f(y, w|x, w) = \sum_w \sum_y f(y, w|x, w) = \sum_w \sum_y f(y, w|x, w_0) = 1$$

from the normalization of f .

In the language of [27], we can see a non-signalling morphism as a degenerate kind of automaton whose future states do not depend on the current states. The causal trace takes a fixed point by reusing identical copies of states.

5.1 Laws of the Causal Trace

We now state and prove a list of properties which the causal trace enjoys. We will compare it with the notion of *partial trace* discussed by [14, 2] and *traced ideal* introduced in [1]. This requires us to postulate a class of traceable morphisms, which we choose as the non-signalling ones, $\mathbb{T}_W^{X,Y} = \{f : X \otimes W \rightarrow Y \otimes W \mid f \text{ is non-signalling}\}$. All proofs are given in the Appendix (Section 9.2).

Proposition 18 (Tightening – naturality in X, Y). If $f : X \otimes W \rightarrow Y \otimes W$ is traceable, then $(h \otimes \text{id}_W) \circ f \circ (g \otimes \text{id}_W) : X' \otimes W \rightarrow Y' \otimes W$ is traceable and

$$\mathbf{tr}^W((h \otimes \text{id}_W) \circ f \circ (g \otimes \text{id}_W)) = h \circ \mathbf{tr}^W(f) \circ g \quad (5)$$

Proposition 19 (Sliding – dinaturality in W). *Let $f : X \otimes W' \rightarrow Y \otimes W$ and $g : W \rightarrow W'$. If f is traceable, then so are $(\text{id}_Y \otimes g) \circ f$ and $f \circ (\text{id}_X \otimes g)$, and*

$$\text{tr}^U((\text{id}_Y \otimes g) \circ f) = \text{tr}^V(f \circ (\text{id}_X \otimes g)) \quad (6)$$

Remark 20. *The corresponding partial trace axiom stipulates that $(\text{id}_Y \otimes g) \circ f$ is traceable if and only if $f \circ (\text{id}_X \otimes g)$ is traceable, but this does not require that traceability of f implies traceability of the composites. Our property is the same as the corresponding traced ideal axiom.*

Proposition 21 (Vanishing (Coherence with I)). *All $g : X \otimes I \rightarrow Y \otimes I$ are traceable, and $\text{tr}^I(g) = g$.*

Proposition 22 (Associativity (Coherence with $\otimes$)). *If $f : X \otimes (U \otimes V) \rightarrow Y \otimes (U \otimes V)$ is traceable, then so are $f : (X \otimes U) \otimes V \rightarrow (Y \otimes U) \otimes V$ and $\text{tr}^V(f) : X \otimes U \rightarrow Y \otimes U$, and*

$$\text{tr}^{U \otimes V}(f) = \text{tr}^U(\text{tr}^V(f))$$

Remark 23. *This property is stronger than the corresponding partial trace axiom, but weaker than the traced ideal axiom.*

Proposition 24 (Superposition (Strength)). *If $f : X \otimes W \rightarrow Y \otimes W$ is traceable, so is $g \otimes f : X' \otimes X \otimes W \rightarrow Y' \otimes Y \otimes W$, and*

$$\text{tr}^W(g \otimes f) = g \otimes \text{tr}^W(f) \quad (7)$$

Proposition 25 (Yanking). *The symmetry is traceable, and*

$$\text{tr}^W(\text{swap}_{W,W}) = \text{id}_W \quad (8)$$

Remark 26. *Our property is the same as the corresponding partial trace axiom. This is stronger than the yanking axiom for traced ideals, as there the symmetry is not required to be traceable.*

6 Causal Traces in Free Markov Categories

In this section, we show that free Markov categories also have causal traces, and that interpretations of Markov string diagrams in cancellative, atomic Markov categories $\mathbf{C}$ with conditionals preserve causal traces. This is an analogue of Theorem 4.2.13 of [16], albeit in a different setting: Markov categories with extra structure instead of universal theories. In particular, we obtain that all contraction identities are satisfied. That is, in the language of [16], the causal trace of a non-signalling morphism in $\mathbf{C}$ can be computed via any stencil representation.

We use free Markov categories as constructed in [12]. Recall that every monoidal signature Σ gives rise to a finite hypergraph which we also denote by Σ . A labelling of wires and boxes in a hypergraph G is a hypergraph homomorphism to Σ . Finite hypergraphs and their homomorphisms form a category **FinHyp**. A *Markov string diagram* is a cospan $\underline{m} \xrightarrow{i} G \xleftarrow{j} \underline{n}$ with discrete hypergraphs $\underline{m}, \underline{n}$, satisfying the conditions of acyclicity, left monogamy, and having no eliminable boxes. The setting is recalled in more detail in Appendix 9.3.

Proposition 27 ([12]). *The free Markov category $\mathbf{FreeMarkov}_\Sigma$ over a monoidal signature Σ can be constructed as follows:*

1. *Objects are hypergraph homomorphisms $\underline{m} \rightarrow \Sigma$, i.e. lists of types in Σ .*
2. *Morphisms are (isomorphism classes of) Markov string diagrams, which compose by pushout and subsequent normalisation (elimination of eliminable boxes).*

3. The tensor is given by coproduct.

To ease notation we will write $f : n \rightarrow m$ for a morphism $f : (\underline{m} \rightarrow \Sigma) \rightarrow (\underline{n} \rightarrow \Sigma)$, thus leaving the labelling implicit. It is straightforward to verify the following:

Proposition 28. *A morphism $f : n \otimes w \rightarrow m \otimes w$ in $\mathbf{FreeMarkov}_\Sigma$ is non-signalling if and only if there are no directed paths from input ports in w to output ports in w .*

We now show that $\mathbf{FreeMarkov}_\Sigma$ has causal traces, by describing the appropriate combinatorial contraction as a contraction of hypergraphs: Let $f : m \otimes w \rightarrow n \otimes w$ be non-signalling, represented by the cospan $\underline{m} + \underline{w} \xrightarrow{[i,i']} G \xleftarrow{[j,j']} \underline{n} + \underline{w}$. Graphically, forming the contraction amounts to gluing the wires connected to matching input and output ports in w , thus making them inner wires (not connected to any input or output port), and normalizing. See Figure 13 for an illustration. The need for normalisation is apparent from the example on Figure 11.

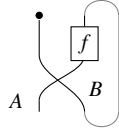


Figure 11: An example where normalisation is needed to compute the trace

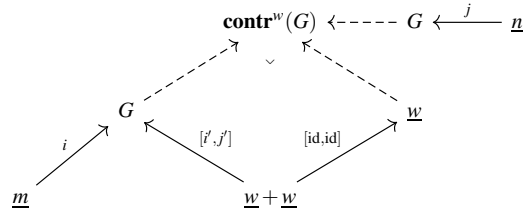


Figure 12: Construction of syntactic traces

More formally, we define the contracted string diagram $\mathbf{contr}^w(f) : m \rightarrow n$ as the normalization of the resulting cospan in the diagram of Figure 12, where the central square is a pushout.

We verify that the resulting string diagram is acyclic and left monogamous. Acyclicity follows from the non-signalling assumption. For left monogamy, observe that after gluing the only affected wires are the one connected to ports in w . Every wire connected to an output port is either also connected to an input port, or is an output of a box. In both cases, left monogamy is preserved, for if a wire was both connected to an input and an output port, the input port cannot be in w by the acyclicity assumption.

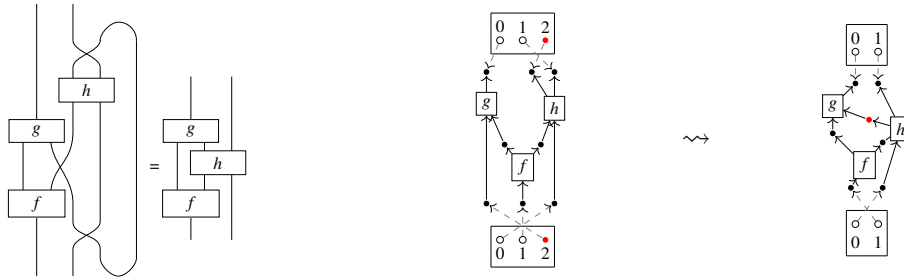


Figure 13: An example of computing the contraction in the hypergraph representation. (Left) The causal trace we want to compute (Right) Hypergraph representations. Black dots represent wires, white dots represent ports; the red ports are being contracted.

Proposition 29. *The contraction operation $\mathbf{contr}$ satisfies the causal trace axioms (as stated in Propositions 18 - 25) for $\mathbf{FreeMarkov}_\Sigma$.*

Proof. Vanishing, strength, and yanking are immediate. The other axioms require bit more caution around the normalization step that occurs in sequential composition and contraction. We can use the fact that normalization of string diagrams is an identity-on-objects gs-monoidal functor $\mathbf{FreeGS}_\Sigma \rightarrow \mathbf{FreeMarkov}_\Sigma$ from the free gs-monoidal category $\mathbf{FreeGS}$ ([12], Lemma 6.5) to solve these cases. $\square$

We finally are ready to formally define our notion of contraction identities.

Definition 30. A Markov category $\mathbf{C}$ is said to satisfy all contraction identities if for all $C_1, C_2 : m \otimes w \rightarrow n \otimes w$ non-signalling in a free Markov category over any signature Σ , and all interpreting functors $\llbracket - \rrbracket : \mathbf{FreeMarkov}_\Sigma \rightarrow \mathbf{C}$, we have that if $\llbracket C_1 \rrbracket = \llbracket C_2 \rrbracket$, then $\llbracket \mathbf{contr}^w(C_1) \rrbracket = \llbracket \mathbf{contr}^w(C_2) \rrbracket$.

This definition is analogous to the ‘notions of contraction’ in [16]. We can show by induction that if $\mathbf{C}$ already has causal traces, then any Markov functor from $\mathbf{FreeMarkov}_\Sigma$ must preserve them:

Proposition 31. Let $\mathbf{C}$ have causal traces, and let $f : m \otimes w \rightarrow n \otimes w$ in $\mathbf{FreeMarkov}_\Sigma$, $\llbracket - \rrbracket : \mathbf{FreeMarkov}_\Sigma \rightarrow \mathbf{C}$ an interpreting functor. Then, if f is non-signalling, so is $\llbracket f \rrbracket$, and $\llbracket \mathbf{contr}^w(f) \rrbracket = \mathbf{tr}^{\llbracket w \rrbracket}(\llbracket f \rrbracket)$.

Proof. The first point is clear. For the second point we proceed by strong induction on w . If $w = 0$, then f is of the form $g \otimes \text{id}_I$. We are done by vanishing (21).

Now assume the statement for all $c < k + 1$. Let $f : n + k + 1 \rightarrow m + k + 1$.

$$\mathbf{tr}^{\llbracket k+1 \rrbracket}(\llbracket f \rrbracket) = \mathbf{tr}^{\llbracket k \rrbracket}(\mathbf{tr}^{\llbracket 1 \rrbracket}(\llbracket f \rrbracket)) = \mathbf{tr}^{\llbracket k \rrbracket}(\llbracket \mathbf{contr}^1(f) \rrbracket) = \llbracket \mathbf{contr}^k(\mathbf{contr}^1(f)) \rrbracket = \llbracket \mathbf{contr}^{k+1}(f) \rrbracket$$

We applied the induction hypothesis twice, and associativity (22) twice. $\square$

As a direct consequence of the previous proposition we obtain:

Corollary 32. Every cancellative atomic Markov category with conditionals satisfies all contraction identities.

7 Notions of Combs

We recall various definitions of combs and refer to [15, 22, 23] for reference.

Definition 33 (Comb). Let $\mathbf{C}$ be a symmetric monoidal category. A comb of type $(A, A') \rightarrow (B, B')$ is a triple $C = \langle f | g \rangle_E$ consisting of an object E and morphisms $f : A \rightarrow E \otimes B$ and $g : E \otimes B' \rightarrow A'$. We will omit the subscript E if it is clear from context. For a morphism $h : B \otimes K \rightarrow B' \otimes K'$, the comb insertion $C[h] : A \otimes K \rightarrow A' \otimes K'$ is defined as $C[h] = (f \otimes \text{id}_K); (\text{id}_E \otimes h); (g \otimes \text{id}_{K'})$. The extension of the comb C is the joint morphism $C[\text{swap}_{B, B'}] : A \otimes B' \rightarrow A' \otimes B$.

Definition 34. Two combs $C_1, C_2 : (A, A') \rightarrow (B, B')$ are called

1. extensionally equivalent if their extensions are equal
2. contextually equivalent if for all $h : B \otimes K \rightarrow B' \otimes K'$, we have $C_1[h] = C_2[h]$
3. optically equivalent if they are identified as elements of the coend

$$\int^E \mathbf{C}(A, E \otimes B) \times \mathbf{C}(E \otimes B', A')$$

Concretely, this is the equivalence generated by ‘sliding’ for all $s : E \rightarrow E'$

$$\langle f; (s \otimes \text{id}_B) | g \rangle_{E'} \sim \langle f | (s \otimes \text{id}_{B'}) ; g \rangle_E$$

Proposition 35 ([15]). *Optic equivalence implies contextual equivalence. Contextual equivalence implies extensional equivalence.*

The converses are false in general. Comb insertion need not be well-defined under extensional equivalence; counterexample atomicity. It is shown in [15] that all three notions coincide in two special cases: compact closed categories and cartesian categories. We generalize this as follows:

Proposition 36. *In any category with causal traces, extensional and contextual equivalence coincide.*

Proof. We can compute the comb insertion from the extension using the causal trace from Figure 13. $\square$

Theorem 37. *Let $\mathbf{C}$ be an atomic Markov category with conditionals. Then extensional and contextual equivalence coincide. Computing the extension defines a bijection between*

1. *morphisms $f : A \otimes B' \rightarrow A' \otimes B$ that are non-signalling*
2. *contextual equivalence classes of combs $(A, A') \rightarrow (B, B')$*

This theorem generalizes the examples covered by [15] to include many common Markov categories without assuming an embedding in a compact-closed category. The case of optic equivalence requires further structure.

Proposition 38. *If $\mathbf{C}$ has universal dilations in the sense of [16], then extensional and optic equivalence coincide.*

Because **FinStoch** has universal dilations ([16, Theorem 2.4.6]), this means that for **FinStoch**, all notions of combs coincide.

8 Conclusions and Future Work

We have shown that the contraction identities hold in every cancellative atomic Markov category with conditionals, and used this to develop a theory of causal traces and relate various notions of comb equivalence. Our work leaves an array of interesting open questions, including about converses of our results:

When does a Markov category embed into a traced category? It is known that every partially traced category embeds in a traced one [2], but our axioms for the causal trace do not imply all partial trace axioms. We conjecture that the difference is similar to what Houghton-Larsen achieves with his construction of *causal channels* in [16, Section 4.1].

To what extent is atomicity a necessary assumption? Does an embedding in a traced category imply atomicity? Atomicity is intimately related to supports [8], and in turn to universal dilations, though the precise relationship remains to be clarified.

Our developments in Section 6 suggest that the information-flow properties of free Markov categories are an interesting area of study beyond this work: We conjecture that these categories are atomic, and have supports but no conditionals. We also believe that free Markov categories embed in free hypergraph categories, even though this point was sidestepped in [12].

We would also like to explore if failure of atomicity poses formal challenges in causal inference, given that the approach to causal inference in [18] relied on combs, or if the approach can be refined to not rely on them.

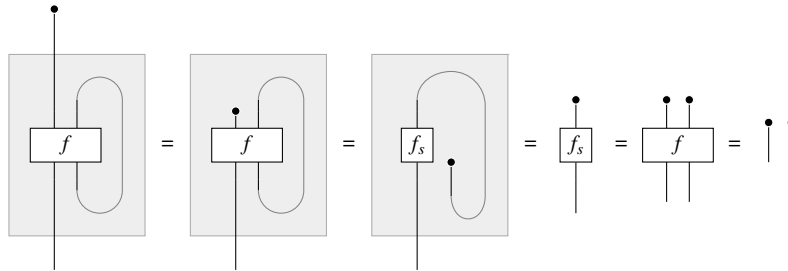
References

- [1] Samson Abramsky, Richard Blute & Prakash Panangaden (1999): *Nuclear and trace ideals in tensored *-categories*. *Journal of Pure and Applied Algebra* 143(1), pp. 3–47, doi:10.1016/S0022-4049(98)00106-6. Available at <https://www.sciencedirect.com/science/article/pii/S0022404998001066>.
- [2] Marc Bagnol (2015): *Representation of Partial Traces*. In Dan R. Ghica, editor: *The 31st Conference on the Mathematical Foundations of Programming Semantics, MFPS 2015, Nijmegen, The Netherlands, June 22-25, 2015, Electronic Notes in Theoretical Computer Science* 319, Elsevier, pp. 37–49, doi:10.1016/J.ENTCS.2015.12.004.
- [3] Giulio Chiribella, G Mauro D’Ariano & Paolo Perinotti (2008): *Quantum circuit architecture*. *Physical review letters* 101(6), p. 060401, doi:10.1103/PhysRevLett.101.060401.
- [4] Kenta Cho & Bart Jacobs (2019): *Disintegration and Bayesian inversion via string diagrams*. *Math. Struct. Comput. Sci.* 29(7), pp. 938–971, doi:10.1017/S0960129518000488.
- [5] Elena Di Lavore & Mario Román (2023): *Evidential Decision Theory via Partial Markov Categories*. In: *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–14, doi:10.1109/LICS56636.2023.10175776.
- [6] Tobias Fritz (2019): *A synthetic approach to Markov kernels, conditional independence, and theorems on sufficient statistics*. *CoRR* abs/1908.07021, doi:10.48550/arXiv.1908.07021. arXiv:1908.07021.
- [7] Tobias Fritz, Tomás Gonda, Nicholas Gauguin Houghton-Larsen, Antonio Lorenzin, Paolo Perrone & Dario Stein (2023): *Dilations and information flow axioms in categorical probability*. *Math. Struct. Comput. Sci.* 33(10), pp. 913–957, doi:10.1017/S0960129523000324.
- [8] Tobias Fritz, Tomáš Gonda, Antonio Lorenzin, Paolo Perrone & Dario Stein (2023): *Absolute continuity, supports and idempotent splitting in categorical probability*. *arXiv preprint arXiv:2308.00651*, doi:10.48550/arXiv.2308.00651.
- [9] Tobias Fritz, Tomáš Gonda & Paolo Perrone (2021): *De Finetti’s Theorem in Categorical Probability*, doi:<https://doi.org/10.48550/arXiv.2105.02639>. arXiv:2105.02639.
- [10] Tobias Fritz, Tomás Gonda, Paolo Perrone & Eigil Fjeldgren Rischel (2023): *Representable Markov categories and comparison of statistical experiments in categorical probability*. *Theor. Comput. Sci.* 961, p. 113896, doi:10.1016/J.TCS.2023.113896.
- [11] Tobias Fritz & Andreas Klingler (2023): *The d-Separation Criterion in Categorical Probability*. *J. Mach. Learn. Res.* 24, pp. 46:1–46:49, doi:10.48550/arXiv.2207.05740. Available at <https://jmlr.org/papers/v24/22-0916.html>.
- [12] Tobias Fritz & Wendong Liang (2023): *Free gs-Monoidal Categories and Free Markov Categories*. *Appl. Categorical Struct.* 31(2), p. 21, doi:10.1007/s10485-023-09717-0.
- [13] Tobias Fritz & Eigil Fjeldgren Rischel (2020): *Infinite products and zero-one laws in categorical probability*. *Compositionality* 2, p. 3, doi:10.32408/COMPOSITIONALITY-2-3.
- [14] Esfandiar Haghverdi & Philip J. Scott (2005): *Towards a Typed Geometry of Interaction*. In C.-H. Luke Ong, editor: *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings, Lecture Notes in Computer Science* 3634, Springer, pp. 216–231, doi:10.1007/11538363_16.
- [15] James Hefford & Cole Comfort (2022): *Coend Optics for Quantum Combs*. In Jade Master & Martha Lewis, editors: *Proceedings Fifth International Conference on Applied Category Theory, ACT 2022, Glasgow, United Kingdom, 18-22 July 2022, EPTCS* 380, pp. 63–76, doi:10.4204/EPTCS.380.4.
- [16] Nicholas Gauguin Houghton-Larsen (2021): *A Mathematical Framework for Causally Structured Dilations and its Relation to Quantum Self-Testing*. Ph.D. thesis, University of Copenhagen, doi:10.48550/arXiv.2103.02302.

- [17] B. Jacobs (2020): *A channel-based perspective on conjugate priors*. *Mathematical Structures in Computer Science* 30(1), p. 44–61, doi:10.1017/S0960129519000082.
- [18] Bart Jacobs, Aleks Kissinger & Fabio Zanasi (2019): *Causal Inference by String Diagram Surgery*. In Mikolaj Bojanczyk & Alex Simpson, editors: *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Lecture Notes in Computer Science* 11425, Springer, pp. 313–329, doi:10.1007/978-3-030-17127-8_18.
- [19] André Joyal, Ross Street & Dominic Verity (1996): *Traced monoidal categories*. *Mathematical Proceedings of the Cambridge Philosophical Society* 119(3), p. 447–468, doi:10.1017/S0305004100074338.
- [20] Aleks Kissinger & Sander Uijlen (2019): *A categorical semantics for causal structure*. *Logical Methods in Computer Science* 15(3), doi:10.23638/LMCS-15(3:15)2019.
- [21] J Pearl (2009): *Causality*. Cambridge university press, doi:10.1017/cbo9780511803161.
- [22] Mario Román (2020): *Comb Diagrams for Discrete-Time Feedback*. *CoRR* abs/2003.06214, doi:10.48550/arXiv.2003.06214. arXiv:2003.06214.
- [23] Mario Román (2020): *Open Diagrams via Coend Calculus*. In David I. Spivak & Jamie Vicary, editors: *Proceedings of the 3rd Annual International Applied Category Theory Conference 2020, ACT 2020, Cambridge, USA, 6-10th July 2020, EPTCS* 333, pp. 65–78, doi:10.4204/EPTCS.333.5.
- [24] P. Selinger (2011): *A Survey of Graphical Languages for Monoidal Categories*, pp. 289–355. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-642-12821-9_4.
- [25] Dario Stein (2021): *Structural Foundations for Probabilistic Programming Languages*. Ph.D. thesis, University of Oxford.
- [26] Dario Stein & Sam Staton (2021): *Compositional Semantics for Probabilistic Programs with Exact Conditioning (long version)*, doi:https://doi.org/10.48550/arXiv.2101.11351. arXiv:2101.11351.
- [27] Nathaniel Virgo (2023): *Unifilar Machines and the Adjoint Structure of Bayesian Models*. arXiv preprint arXiv:2305.02826, doi:10.48550/arXiv.2305.02826.

9 Appendix

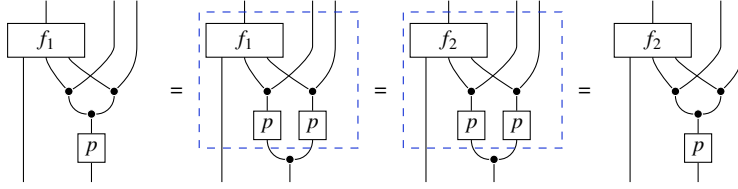
Proof of Proposition 16. The trace of a discardable morphism f need not itself be discardable. For example, in the compact closed category $\mathbf{Mat}(\mathbb{R}^+)$, the trace of $\text{id}_W : W \rightarrow W$ is the scalar $|W| : I \rightarrow I$ which is not normalized (i.e. equal to 1). On the other hand, if f is non-signalling, we obtain



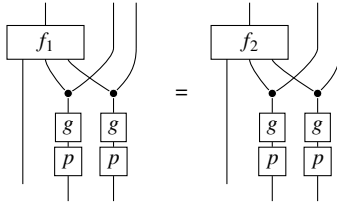
□

9.1 Appendix to Section 4

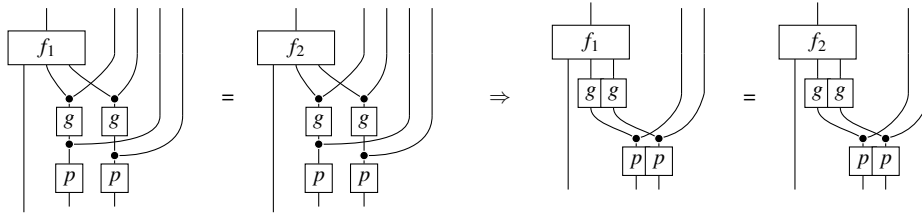
Proof of Proposition 9. The statement for morphisms of full support is immediate. For a deterministic morphism p , we reason straightforwardly



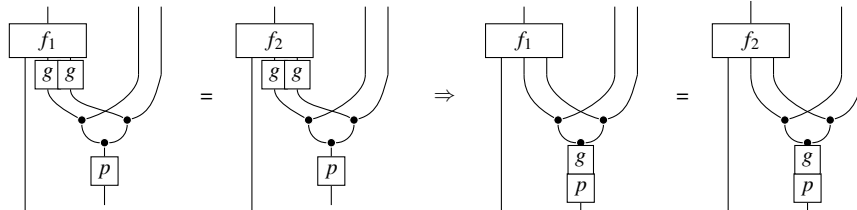
For the third point, let g be deterministic and p atomic, and assume



By causality [7, A.14], we may strengthen this equation and marginalize to obtain



Now, we use atomicity of p , postcompose the two rightmost wires with g and use determinism to simplify



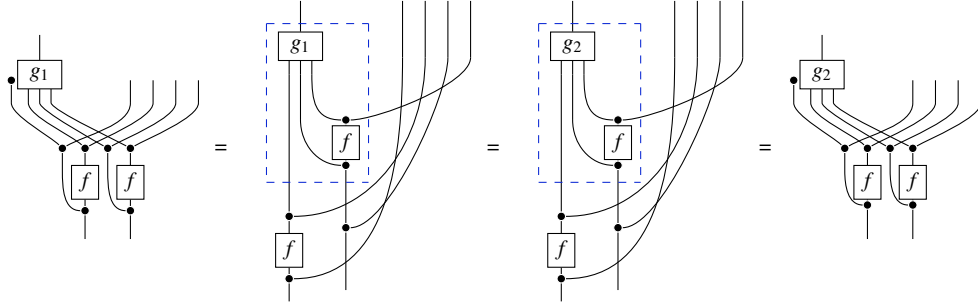
□

Example 39. Atomic morphisms need not be closed under composition: In **BorelStoch**, let $X = [0, 1]$ and define the morphism $p : X \otimes 2 \rightarrow X$ by

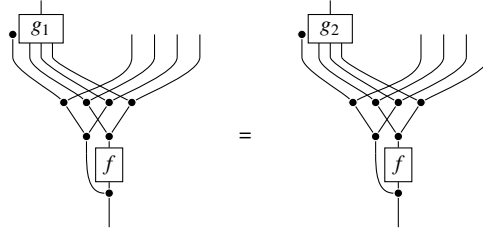
$$p(x, c) = \begin{cases} \delta_x, & c = 0 \\ \mathbf{v}, & c = 1 \end{cases}$$

Then the atoms of p are all of X , i.e. p is atomic. However, for all deterministic states $\sigma : I \rightarrow X \otimes 2$ of the form $\sigma = \delta_{(x, 1)}$, we have that $p \circ \sigma = \mathbf{v}$ is not atomic.

Proof of Lemma 11. First, by marginalizing Y and cancellability of W , we conclude that $f_1 = f_2$ and write f indiscriminately. Now we show that the morphisms $c_i = \text{del}_X \otimes g_i$ (for $i = 1, 2$) are $\phi \otimes \phi$ -almost surely equal, where $\phi = (\text{id}_X \otimes f) \circ \Delta_X$.

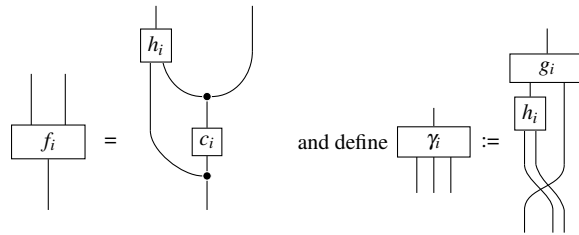


Applying atomicity to ϕ , we obtain that c_1, c_2 are also $(\Delta \circ \phi)$ -almost surely equal, i.e.

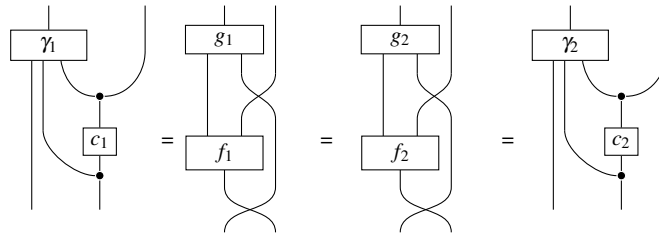


From this, we obtain the desired conclusion by marginalizing the four wires on the right. $\square$

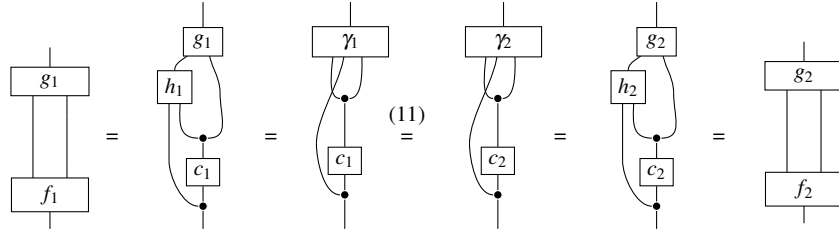
Proof of Lemma 12. For $i = 1, 2$, disintegrate f_i as



Then



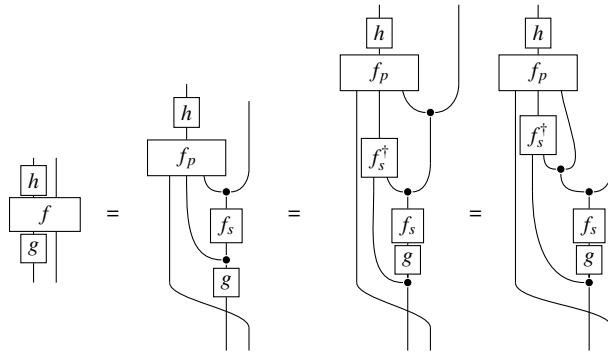
So by Lemma 11, we conclude



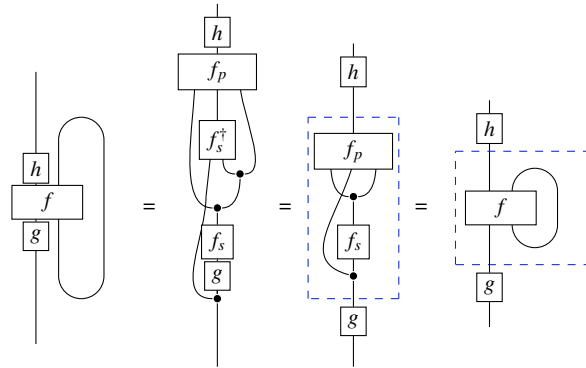
□

9.2 Appendix to Section 5.1

Proof of Tightening (Proposition 18). Write k for the composite $(h \otimes \text{id}_W) \circ f \circ (g \otimes \text{id}_W)$; it is easy to see that k is traceable. Using the Bayesian inverse $(f_s)^\dagger_g$, we obtain the following disintegration



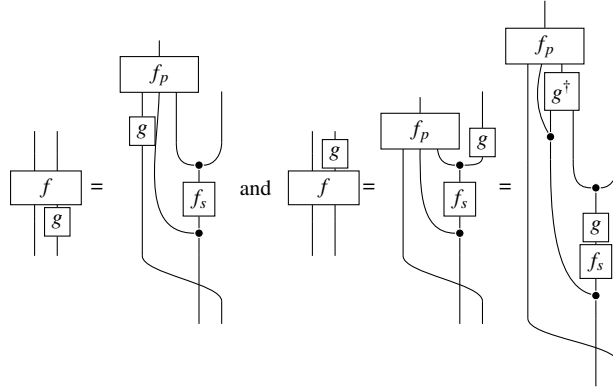
Using that disintegration, we compute as desired



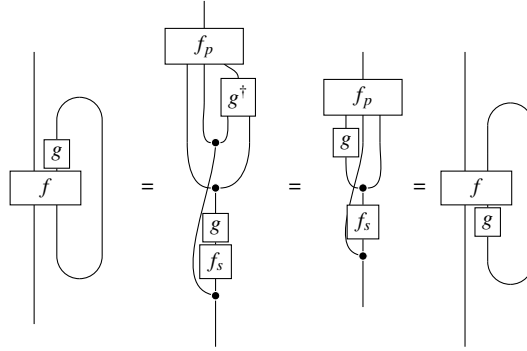
□

Proof of Sliding (Proposition 19). Traceability is straightforward. Using the Bayesian inverse $g_{f_s}^\dagger$, we

establish the following disintegrations:

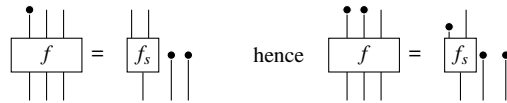


Using these disintegrations, we show

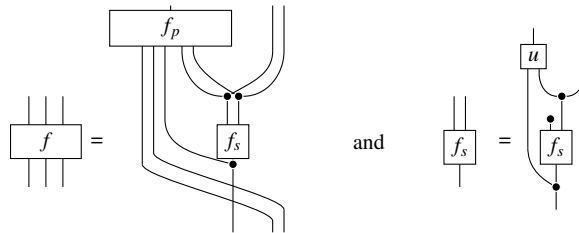


□

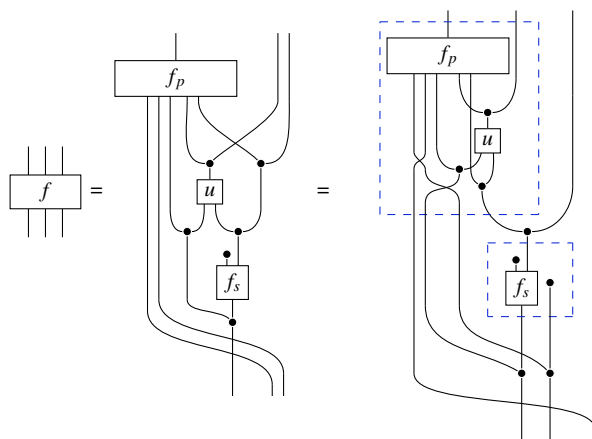
Proof of Associativity (Proposition 22). Assume that $f : X \otimes (U \otimes V) \rightarrow Y \otimes (U \otimes V)$ is non-signalling in $U \otimes V$; that is



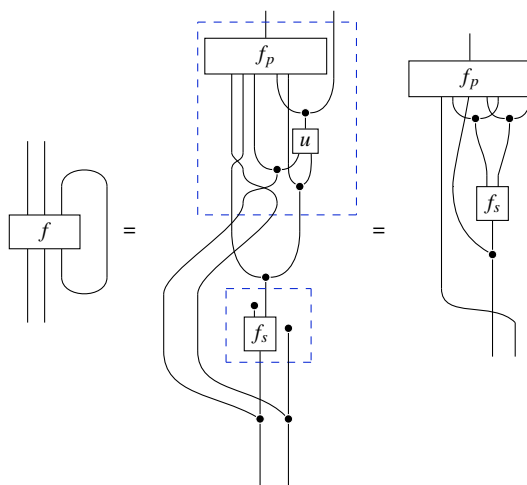
hence f is in particular non-signalling from V to V . Choose a disintegration f_p, f_s , and condition f_s further, to obtain



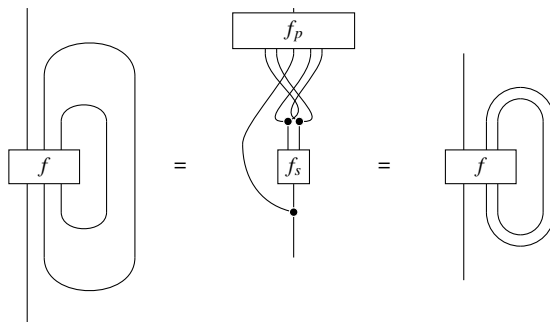
Then we can give the following disintegration with respect to V



Using this disintegration, we obtain $\mathbf{tr}^V(f)$ as

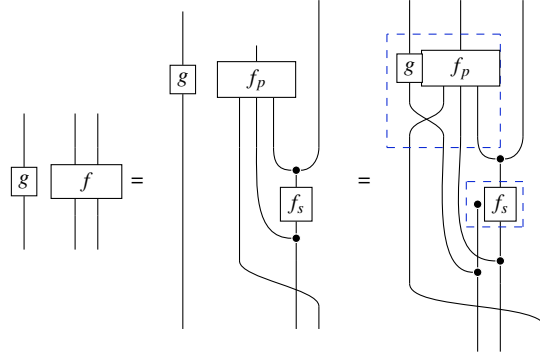


Of this, we can in turn compute the trace

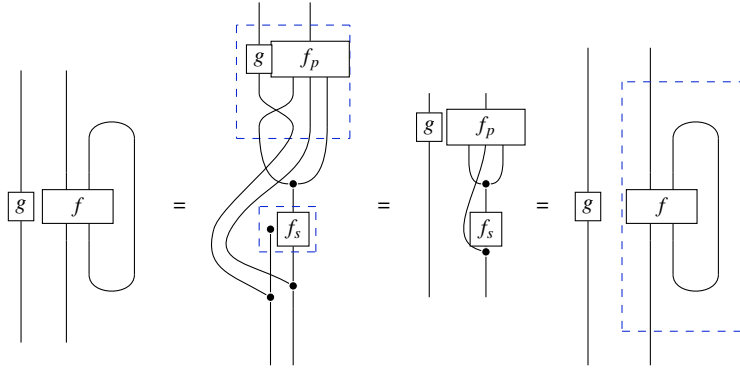


□

Proof of Superposition (Proposition 24). We make use of the following disintegration

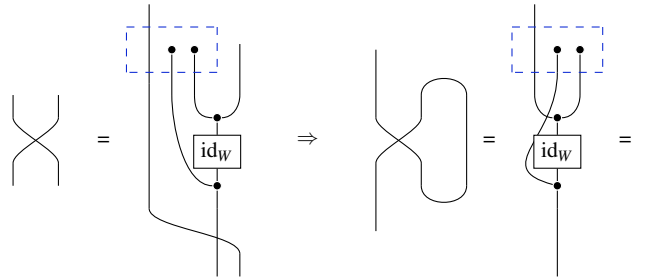


to obtain



□

Proof of Yanking (Proposition 25). The symmetry has the following disintegration, hence



□

9.3 Appendix to Section 6

We recall the construction of free Markov categories [12].

Definition 40 ([12]). *The category $\mathbf{I}$ is defined as follows.*

- *Objects are pairs of natural numbers $(m, n) \in \mathbb{N} \times \mathbb{N}$, and there is an extra object $*$.*
- *The only non-identity morphisms are $\text{in}_1, \dots, \text{in}_m, \text{out}_1, \dots, \text{out}_n : (m, n) \rightarrow *$.*

A hypergraph is a functor $\mathbf{I} \rightarrow \mathbf{Sets}$. Hypergraphs form a functor category $\mathbf{Hyp}$.

We also use the following notation for a hypergraph $G : \mathbf{I} \rightarrow \mathbf{Sets}$.

- $W(G) = G(*)$ is the set of wires.
- $B_{m,n}(G)$ is the set of boxes with m inputs and n outputs. $B(G) = \bigsqcup_{m,n \in \mathbb{N}} B_{m,n}(G)$ is the set of all boxes.
- We abbreviate $G(\text{in}_i)$ to in_i and $G(\text{out}_i)$ to out_i . These assign the i th input/output wire to each box.
- For $b \in B_{m,n}(G)$, $w \in W(G)$

$$\begin{aligned} \text{in}(b, w) &= \{\text{in}_i(b) : i \in \{1, \dots, m\}\} \\ \text{out}(b, w) &= \{\text{out}_i(b) : i \in \{1, \dots, n\}\} \end{aligned}$$

These numbers tell how many times a wire is the input/output of a box.

Definition 41 ([12]). A hypergraph is finite if the set of wires $W(G)$ and the set of boxes $B(G)$ is finite. We denote the subcategory of finite hypergraphs by **FinHyp**.

Every monoidal signature Σ gives rise to a finite hypergraph which we also denote by Σ . A labelling of wires and boxes in a hypergraph G is a hypergraph homomorphism (natural transformation) to Σ .

Definition 42 ([12]). Markov string diagrams over the monoidal signature Σ are (isomorphism classes of) cospans in the slice category **FinHyp**/ Σ , that is of the form

$$\begin{array}{ccc} \underline{m} & \xrightarrow{p} & G \xleftarrow{q} \underline{n} \\ & \searrow & \downarrow \swarrow \\ & & \Sigma \end{array}$$

in **FinHyp**, such that

1. $\underline{m}$ and $\underline{n}$ are discrete hypergraphs ($B(\underline{m}) = B(\underline{n}) = \emptyset$).
2. G is acyclic, i.e. it contains no cycles. A path is a finite, alternating sequence of wires and boxes $(w_1, b_1, \dots, w_n, b_n, w_{n+1})$ such that $\text{in}(b_i, w_i) > 0$, and $\text{out}(b_i, w_{i+1}) > 0$ for all $1 \leq i \leq n$. A cycle is a path that additionally satisfies $w_1 = w_{n+1}$.
3. The cospan satisfies left monogamy: for all wires $w \in W(G)$

$$|p^{-1}(w)| + \sum_{b \in B(G)} \text{out}(b, w) = 1$$

That is, every wire is either connected to exactly one input port or is the output of a single box.

4. There are no eliminable boxes. A box is eliminable if none of its output wires are connected to an output port or to the input of a box. That is, an eliminable box b satisfies for all $w \in W(G)$

$$\text{out}(b, w) > 0 \implies q^{-1}(w) = 0 \wedge \forall b' \in B(G). \text{in}(b', w) = 0$$

We leave the labelling implicit and write the cospan as $\underline{m} \xrightarrow{p} G \xleftarrow{q} \underline{n}$.

Proposition 43 ([12]). The free Markov category **FreeMarkov** $_{\Sigma}$ over a monoidal signature Σ can be constructed as follows:

1. Objects are homomorphisms $\underline{m} \rightarrow \Sigma$, that is an m -long list of types in Σ .
2. Morphisms are (isomorphism classes of) Markov string diagrams, which compose by pushout and subsequent normalisation (elimination of eliminable boxes).

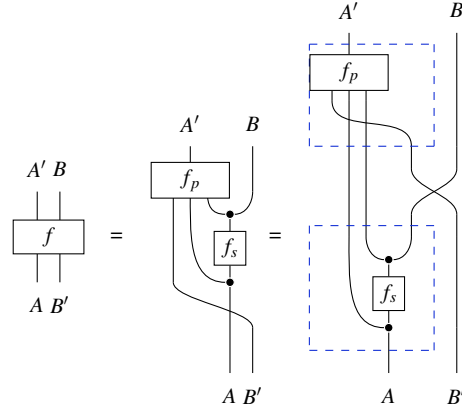
3. The tensor is given by coproduct.
4. Copy and delete are represented by the cospans

$$\boxed{0 \bullet} \triangleright \bullet \boxed{\bullet 0} \quad \boxed{0 \bullet} \triangleright \bullet \boxed{\bullet} \quad \boxed{\bullet} \triangleright \bullet \boxed{\bullet}$$

To ease notation we write $f : n \rightarrow m$ for a morphism $f : (\underline{m} \rightarrow \Sigma) \rightarrow (\underline{n} \rightarrow \Sigma)$, thus leaving the labelling implicit.

9.4 Appendix to Section 7

Proof of Theorem 37. For each comb $\langle f|g \rangle : (A, A') \rightarrow (B, B')$, its extension is a non-signalling morphism $A \otimes B' \rightarrow A' \otimes B$. Conversely, from a non-signalling morphism $f : A \otimes B' \rightarrow A' \otimes B$, we construct a disintegration



which defines a comb with environment $E = A \otimes B$ whose extension is f . By Proposition 36, the resulting comb is unique up to contextual equivalence. $\square$

Universal Dilations In the terminology of [16, Definition 2.4.1], a *universal dilation* of a morphism $p : X \rightarrow Y$ is another morphism $\phi : X \rightarrow E \otimes Y$ satisfying

$$\begin{array}{c} Y \\ \bullet \\ \boxed{\phi} \\ X \end{array} \quad E = \begin{array}{c} Y \\ \boxed{p} \\ X \end{array}$$

such that for all $\Pi : X \otimes W' \rightarrow W \otimes Y$

$$\text{if } \begin{array}{c} Y \\ \bullet \\ \boxed{\Pi} \\ X \quad W' \end{array} = \begin{array}{c} Y \\ \boxed{p} \\ X \quad W' \end{array} \quad \text{then } \exists! \gamma \text{ with } \begin{array}{c} \boxed{\Pi} \end{array} = \begin{array}{c} \boxed{\gamma} \\ \boxed{\phi} \end{array}$$

Proof of Proposition 38. Let $\langle f_1 | g_1 \rangle_{E_1}, \langle f_2 | g_2 \rangle_{E_2} : (A, A') \rightarrow (B, B')$ be two combs which are extensionally equivalent. Define using cancellability $p : A \rightarrow B$ as the common morphism

$$\begin{array}{c} \bullet \\ | \\ \boxed{f_1} \end{array} = \begin{array}{c} | \\ \boxed{p} \end{array} = \begin{array}{c} \bullet \\ | \\ \boxed{f_2} \end{array}$$

and choose a universal dilation $\phi : A \rightarrow E \otimes B$ of p . Using universality, the f_i must factor through ϕ as

$$\begin{array}{c} | \\ | \\ \boxed{f_i} \end{array} = \begin{array}{c} | \\ \boxed{r_i} \\ | \\ \boxed{\phi} \end{array}$$

By uniqueness of factorization, we have

$$\begin{array}{c} | \\ \boxed{g_1} \\ | \\ \boxed{r_1} \\ | \\ \boxed{\phi} \end{array} = \begin{array}{c} | \\ \boxed{g_2} \\ | \\ \boxed{r_2} \\ | \\ \boxed{\phi} \end{array} \quad \text{hence} \quad \begin{array}{c} | \\ \boxed{g_1} \\ | \\ \boxed{r_1} \end{array} = \begin{array}{c} | \\ \boxed{g_2} \\ | \\ \boxed{r_2} \end{array}$$

Now we reason modulo optic equivalence that

$$\begin{array}{c} | \\ \boxed{g_1} \\ | \\ \boxed{f_1} \end{array} = \begin{array}{c} | \\ \boxed{g_1} \\ | \\ \boxed{r_1} \\ | \\ \boxed{\phi} \end{array} \sim \begin{array}{c} | \\ \boxed{g_1} \\ | \\ \boxed{r_1} \\ | \\ \boxed{\phi} \end{array} = \begin{array}{c} | \\ \boxed{g_2} \\ | \\ \boxed{r_2} \\ | \\ \boxed{\phi} \end{array} \sim \begin{array}{c} | \\ \boxed{g_2} \\ | \\ \boxed{r_2} \\ | \\ \boxed{\phi} \end{array} = \begin{array}{c} | \\ \boxed{g_2} \\ | \\ \boxed{f_2} \end{array}$$

□

Presenting Profunctors

Gabriel Goren-Roig

Universidad de Buenos Aires
CONICET
ggoren@dc.uba.ar

Joshua Meyers

Conexus AI
meygerjos@protonmail.com

Emilio Minichiello

CUNY CityTech
eminichiello67@gmail.com

Motivated by problems in categorical database theory, we introduce and compare two notions of presentation for profunctors, uncurried and curried, which arise intuitively from thinking of profunctors either as functors $\mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$ or $\mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}^{\mathcal{D}}$. Although the Cartesian closure of $\mathbf{Cat}$ means these two perspectives can be used interchangeably at the semantic level, a surprising amount of subtlety is revealed when looking through the lens of syntax. Indeed, we prove that finite uncurried presentations are strictly more expressive than finite curried presentations, hence the two notions do not induce the same class of finitely presentable profunctors. Moreover, an explicit construction for the composite of two curried presentations shows that the class of finitely curried presentable profunctors is closed under composition, in contrast with the larger class of finitely uncurried presentable profunctors, which is not. This shows that curried profunctor presentations are more appropriate for computational tasks that use profunctor composition. We package our results on curried profunctor presentations into a double equivalence from a syntactic double category into the double category of profunctors. Finally, we study the relationship between curried and uncurried presentations, leading to the introduction of curryable presentations. These constitute a subcategory of uncurried presentations which is equivalent to the category of curried presentations, therefore acting as a bridge between the two syntactic choices.

Acknowledgements: We thank the reviewers for their detailed comments and suggestions. We also thank David Spivak and Ryan Wisnesky for helpful conversations.

1 Introduction

In the most recent iteration of categorical database theory [Spi12; SW15; PLF22; Sch+17; SW17], profunctors are pervasive. Indeed, they are used to define database schemas with attributes as well as to define generalized queries on such schemas, which, fundamentally, can also implement data exchange operations. In this context, composition of profunctors plays a key role, since it allows us to define composite queries in a data-independent way, and it specializes to *query coevaluation* [Sch+17], a natural categorical operation on database instances which may have uses in data integration.

In order to encode these entities as finite data and define algorithms on them that implement operations such as composition, we must move into the territory of finite presentations of these mathematical objects, and, as we will show, different perspectives on profunctors naturally suggest distinct, inequivalent notions of presentation. In [SW17], the authors develop a syntactic theory of data integration based on the concept of *uberflowers*, which can be thought of as a notion of presentation for a certain generalization of profunctors. Note that uberflowers are central to the database algorithms implemented in the open-source CQL tool available at categoricaldata.net. When taking the particular case of profunctors, this syntactic notion reduces to what we call *curried profunctor presentations*. These are natural when thinking of profunctors $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ as functors $\mathcal{P} : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}^{\mathcal{D}}$.

However, existing work on uberflowes does not connect these syntactic entities with their semantics as (generalized) profunctors. On the contrary, although the semantic counterpart of the theory of uberflowes is developed in [Sch+17], there the authors take the alternative view of profunctors as functors of the form $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$. From this perspective, they define a notion of presentation for their semantic version of queries, which they call *bimodule presentation*, and which reduces in the case of profunctors to what we call *uncurried profunctor presentations*.

Owing to the Cartesian closure of **Cat**, these two descriptions of a profunctor are interchangeable. However, we will show that the corresponding syntactic notions of curried and uncurried profunctor presentations are not, since they give rise to inequivalent notions of finite presentability, and hence *a fortiori* uberflowes and bimodule presentations are not equivalent notions.

In Section 2, we discuss presentations of small categories. In Section 3, after having reviewed profunctors, we introduce our main objects of study: uncurried and curried profunctor presentations. We begin our exposition with uncurried profunctor presentations (Definition 3.4) and prove (Proposition 3.9) that the class of finitely uncurried presentable profunctors is not closed under composition. This already implies that bimodule presentations as defined in [Sch+17] are not suitable for implementation.

We then derive presentations for copresheaves on a category presentation C , which we call *C-instance presentations*, as a particular case of uncurried profunctor presentations (Definition 3.11). This is done in order to finally define curried presentations as indexed families of instance presentations and morphisms between them (Definition 3.15). We define a composition operation for curried profunctor presentations and prove its correctness (Theorem 3.26), which constitutes one of our main results. This immediately implies that the class of finitely curried presentable profunctors is closed under composition. We conclude that curried presentations are indeed appropriate for computing profunctor composition. Although the idea behind composition of curried profunctor presentations is present in [SW17], the precise mathematical formulation and the proof of correctness are novel to the best of our knowledge.

In Section 4, we turn towards the question of how exactly these two notions of profunctor presentation relate to each other. We begin by noting that there is an evident “uncurrying” functor from curried to uncurried presentations which preserves the semantics and sends finite presentations to finite presentations (Proposition 4.2). This, together with the fact that there are finitely uncurried presentable profunctors which are not finitely curried presentable (Lemma 3.20), lets us conclude that finite uncurried presentations are strictly more expressive than their curried counterparts. We then study the image of the uncurrying functor, and in doing so we arrive at the concept of *curryable profunctor presentations*. In our second main theorem (Theorem 4.11), we prove that the resulting category of curryable profunctor presentations is equivalent to the category of curried presentations, and that this equivalence restricts to the respective full subcategories of finite presentations. This shows that we can indeed replace finite curried presentations by finite curryable presentations, which may be more convenient to work with.

Finally, in Section 5 we package our results on curried profunctor presentations into a double functor $(-)\downarrow : \mathbf{Curr} \rightarrow \mathbf{Prof}$ of double categories, and show that by appropriately quotienting the domain of this double functor, we obtain a double equivalence.

For clarity and space considerations, we defer most proofs of this paper to the Appendix.

2 Category Presentations

In this section, we define category presentations together with their semantics via a functor $(-)\downarrow : \mathbf{CatPr} \rightarrow \mathbf{Cat}$. This functor factors through an appropriate quotient $\mathbf{CatPr}_{\approx}$ of $\mathbf{CatPr}$ and induces an equivalence $\mathbf{CatPr}_{\approx} \simeq \mathbf{Cat}$. We follow the terminology of algebraic theories (sorts, function symbols and signatures).

Similar definitions can be found in [Spi14].

Definition 2.1. A **category signature** Σ consists of a set $\text{Sort}(\Sigma)$, whose elements we call sorts, and a set $\text{Fun}(\Sigma)$, whose elements we call function symbols, along with two functions $s, t : \text{Fun}(\Sigma) \rightarrow \text{Sort}(\Sigma)$, called the source and target maps. We write $f : c_0 \rightarrow c_1$ to denote that $s(f) = c_0$ and $t(f) = c_1$.

Given a category signature Σ and $c, c' \in \text{Sort}(\Sigma)$, a **path in Σ from c to c'** , denoted as $p : c \rightarrow c'$, consists of a (possibly empty) finite, ordered list of function symbols $p = (f_0, f_1, \dots, f_{n-1})$, such that if it is nonempty then $s(f_0) = c$, $t(f_{n-1}) = c'$ and $t(f_i) = s(f_{i+1})$ for all $0 \leq i < n-1$, and if it is empty we require $c = c'$. In the first case we write $p = f_0.f_1 \dots f_{n-1}$ while in the second case we write $p = 1_c$. Let $\text{Path}(\Sigma)(c, c')$ denote the set of paths in Σ from c to c' and $\text{Path}(\Sigma)$ denote the set of all paths.

Given a category signature Σ , an **equation** over Σ consists of a pair (p_0, p_1) where $p_0, p_1 : c \rightarrow c'$ are paths in Σ with the same source and target. We denote such an equation by $p_0 = p_1 : c \rightarrow c'$.

A **category presentation** is a pair $C = (C_\Sigma, C_E)$ of a category signature C_Σ and a set C_E of equations over C_Σ . We say that C is a **finite category presentation** if $\text{Sort}(C_\Sigma)$, $\text{Fun}(C_\Sigma)$ and C_E are all finite sets.

Notation 2.2. If $C = (C_\Sigma, C_E)$ is a category presentation, then we let $\text{Sort}(C) := \text{Sort}(C_\Sigma)$, $\text{Fun}(C) := \text{Fun}(C_\Sigma)$, and $\text{Path}(C) := \text{Path}(C_\Sigma)$. We write $c \in C$ to mean that $c \in \text{Sort}(C)$. Given $c, c' \in C$, we write $\text{Fun}(C)(c, c')$ to denote the subset of function symbols f of C with $s(f) = c$ and $t(f) = c'$. We write $p_0 =_C p_1$ to mean that the equation $p_0 = p_1$ belongs to C_E . Since the equality symbol is taken up for equations in category presentations, given two paths p and q we use the notation $p \equiv q$ to indicate true (syntactic) equality. We still use $=$ for true equality of sorts and semantic objects.

Equations in a category presentation constitute axioms from which equivalence of paths may be deduced by the rules of equational logic. This is captured by the following definition.

Definition 2.3. Given a category presentation C , define a relation $\approx_C$ on paths of C inductively¹ by the following inference rules:

1. For any equation $p =_C q$, $p \approx_C q$.
2. For any path p , $p \approx_C p$.
3. If $p \approx_C q$, then $q \approx_C p$.
4. If $p \approx_C q$ and $q \approx_C r$, then $p \approx_C r$.
5. For any C -function symbol $f : c' \rightarrow c''$, if $p \approx_C q : c \rightarrow c'$, then $p.f \approx_C q.f$.
6. For any C -function symbol $f : c \rightarrow c'$, if $p \approx_C q : c' \rightarrow c''$, then $f.p \approx_C f.q$.

We call $\approx_C$ the **C -provable equality relation** on paths, and we say that p_0 and p_1 are C -provably equal if $p_0 \approx_C p_1$.

We now define morphisms between category presentations. These map function symbols to paths in a way that sends equations (axioms) to provable equalities (equational theorems obtained from those axioms).

Definition 2.4. Given category signatures Σ and Σ' , a **morphism of category signatures** $F : \Sigma \rightarrow \Sigma'$ consists of functions $F_{\text{Sort}} : \text{Sort}(\Sigma) \rightarrow \text{Sort}(\Sigma')$ and $F_{\text{Fun}} : \text{Fun}(\Sigma) \rightarrow \text{Path}(\Sigma')$, such that if $f : c \rightarrow c'$ is a function symbol in Σ , then $F_{\text{Fun}}(f) : F_{\text{Sort}}(c) \rightarrow F_{\text{Sort}}(c')$. We refer to both F_{Sort} and F_{Fun} by F . We extend F to $\text{Path}(\Sigma)$ by setting $F(f_0.f_1 \dots f_{n-1}) := F(f_0).F(f_1) \dots F(f_{n-1})$ and $F(1_c) := 1_{F(c)}$.

Given category presentations C and C' , a **morphism of category presentations** $F : C \rightarrow C'$ is a morphism $F : C_\Sigma \rightarrow C'_\Sigma$ of category signatures such that for each equation $p_0 =_C p_1$, we have $F(p_0) \approx_{C'} F(p_1)$. Let **CatPr** denote the category of category presentations. Notice that the action of morphisms of category presentations on arbitrary paths is necessary for composition to be well defined.

¹That is to say, there are no pairs (p, q) in $\approx_C$ other than what can be derived in finitely many steps with the rules 1-6.

We obtain a category out of a category presentation C by performing a quotient of the category freely generated by C_Σ according to C_E . To that end we must first define congruences of categories (see [Bor94, Section 5.1] and [BBP99]).

Definition 2.5. Let $\mathcal{C}$ be a small category, and $\sim$ an equivalence relation on $\text{Mor}(\mathcal{C})$. We say that $\sim$ is a **congruence on $\mathcal{C}$** if whenever $f \sim g$, f and g share the same domain and codomain; and furthermore, if $f_1 \sim f_2$, $g_1 \sim g_2$ and the composition $g_1 \circ f_1$ exists, then $g_1 \circ f_1 \sim g_2 \circ f_2$. If f is a morphism in $\mathcal{C}$, then we denote its $\sim$ -equivalence class by $[f]$.

If $\sim$ is a congruence on $\mathcal{C}$, let $\mathcal{C}/\sim$ denote the small category whose objects are the same as $\mathcal{C}$, whose set of morphisms is $\text{Mor}(\mathcal{C})/\sim$, and where we define $1_c := [1_c]$ and $[g] \circ [f] := [g \circ f]$. We call it the **quotient category** of $\mathcal{C}$ by $\sim$ and denote it by $\mathcal{C}/\sim$.

If R is a relation on $\text{Mor}(\mathcal{C})$ such that whenever fRg , f and g share the same domain and codomain, let $\sim_R$ be the smallest congruence that contains R . Then we define $\mathcal{C}/R := \mathcal{C}/\sim_R$, and we let $p_R : \mathcal{C} \rightarrow (\mathcal{C}/R)$ denote the canonical projection functor.

Lemma 2.6. Let $\mathcal{C}$ be a small category and R be a relation on $\text{Mor}(\mathcal{C})$ such that whenever fRg , f and g share the same domain and codomain. Suppose that $F : \mathcal{C} \rightarrow \mathcal{D}$ is a functor with the property that if f and g are morphisms in $\mathcal{C}$ such that fRg , then $F(f) = F(g)$. Then there exists a unique functor $\tilde{F} : (\mathcal{C}/R) \rightarrow \mathcal{D}$ such that $\tilde{F} \circ p_R = F$.

Lemma 2.7. Given a category presentation C , $\approx_C$ is the smallest congruence containing C_E . In other words, the relations $\sim_{C_E}$ and $\approx_C$ coincide.

Definition 2.8. Given a category signature Σ , let $\text{Free}(\Sigma)$ denote the small category whose objects are the sorts of Σ and whose homsets are given by $\text{Hom}(c, c') := \text{Path}(c, c')$.

Given a category presentation C , let $\langle C \rangle := \text{Free}(C_\Sigma)/C_E = \text{Free}(C_\Sigma)/\approx_C$. We call $\langle C \rangle$ the **category presented by C** . If $p : c \rightarrow c'$ is a C -path, then we let $[p] \in \langle C \rangle(c, d)$ denote its corresponding $\approx_C$ -equivalence class in $\langle C \rangle$.

Although we are interested in general category presentations (together with instance and profunctor presentations on them, as we will see in the subsequent sections), for examples and counterexamples we will concentrate on the particular case in which category presentations have only one sort $*$, in which case we call them **monoid presentations**. Much of the subtlety of the theory is already present in this class of examples, and it frees us from having to keep track of sorts.

Example 2.9. Given a monoid presentation M , we write $M = \langle \text{Fun}(M) \mid M_E \rangle$ following the usual notation for e.g. group presentations. For instance, let $M = \langle f, g \mid f.g = g.f \rangle$. Then $\langle M \rangle$ is the free commutative monoid on two generators. Moreover, let $F : M \rightarrow M$ be defined by $F(f) := f.f$, $F(g) := f.g$. This is a well defined morphism of category presentations because $F(f.g) \equiv f.f.f.g \approx_M f.g.f.f \equiv F(g.f)$.

We use the notation $\langle \text{Fun}(M) \mid \emptyset \rangle$ to denote monoid presentations with no equations, and $\langle \emptyset \mid \emptyset \rangle$ to denote trivial monoid presentations, with no function symbols or equations. Notice that since this notation omits the unique sort, it technically only defines a presentation up to unique isomorphism. We usually denote the unique sorts as $*$ unless specified otherwise.

We have made use of congruences of categories in order to define the semantics of category presentations. We will now use them a second time in order to define a quotient of the “bigger” category **CatPr**.

Lemma 2.10. Given a morphism $F : C \rightarrow C'$ of category presentations, if $p_0 \approx_C p_1$, then $F(p_0) \approx_{C'} F(p_1)$.

Definition 2.11. We say that two morphisms $F, G : C \rightarrow D$ of category presentations are **provably equal**, and write $F \approx G$, if and only if for each $c \in \text{Sort}(C)$, the sorts $F(c)$ and $G(c)$ are equal and for each $f \in \text{Fun}(C)$, $F(f) \approx_D G(f)$. It is easily shown using Lemma 2.10 that the relation $\approx$ is a congruence on $\mathbf{CatPr}$, so we define $\mathbf{CatPr}_{\approx} := \mathbf{CatPr} / \approx$.

The construction $\langle - \rangle$ from Definition 2.8 extends to a functor $\langle - \rangle : \mathbf{CatPr} \rightarrow \mathbf{Cat}$. This functor sends provably equal morphisms to equal morphisms, so it induces a unique functor $\langle - \rangle : \mathbf{CatPr}_{\approx} \rightarrow \mathbf{Cat}$ by Lemma 2.6. In what follows we will abuse notation and allow $\langle - \rangle$ to also denote $\langle - \rangle$.

Proposition 2.12. The functor $\langle - \rangle : \mathbf{CatPr}_{\approx} \rightarrow \mathbf{Cat}$ described above is an equivalence of categories.

Proof sketch. Given a small category $\mathcal{C}$, let $\text{Can}(\mathcal{C})$ denote its canonical category presentation², whose sorts are the objects of $\mathcal{C}$, whose function symbols are the morphisms of $\mathcal{C}$, and whose set of equations is the set of pairs of paths in $\text{Can}(\mathcal{C})$ whose compositions in $\mathcal{C}$ are equal. This construction can be shown to extend to a functor $\text{Can} : \mathbf{Cat} \rightarrow \mathbf{CatPr}_{\approx}$ which is a quasi-inverse for $\langle - \rangle$. $\square$

Note that $\mathbf{CatPr}$ is not equivalent to $\mathbf{Cat}$, as the following example shows. On the contrary, equality of morphisms in $\mathbf{CatPr}$ is a strict, purely syntactic notion.

Example 2.13. Let $N := \langle s \mid \emptyset \rangle$ and $M := \langle f, g \mid f.g = g.f \rangle$ denote monoid presentations. Let $F, G : N \rightarrow M$ denote morphisms of category presentations defined by $F(s) := f.g$ and $G(s) := g.f$. Now $\langle F \rangle([s]) = [f.g] = [g.f] = \langle G \rangle([s])$, so $\langle F \rangle = \langle G \rangle$, but $F \neq G$ as morphisms in $\mathbf{CatPr}$, so $\langle - \rangle$ is not faithful.

3 Profunctor Presentations

In this section we introduce the notions of uncurried and curried profunctor presentations. First we recall the definition of profunctors and profunctor composition.

Definition 3.1. Let $\mathbf{2}$ denote the category with two objects 0 and 1, and a single non-identity morphism $\leq : 0 \rightarrow 1$. Given categories $\mathcal{C}$ and $\mathcal{D}$, a **profunctor** $\mathcal{P} : \mathcal{C} \nrightarrow \mathcal{D}$ is a category $\mathcal{P}$ equipped with a functor $\pi : \mathcal{P} \rightarrow \mathbf{2}$, such that $\pi^{-1}(0) = \mathcal{C}$ and $\pi^{-1}(1) = \mathcal{D}$. We say a morphism f in $\mathcal{P}$ is a **cross-morphism** if $\pi(f) = \leq$. Given profunctors $\mathcal{P}, \mathcal{P}' : \mathcal{C} \nrightarrow \mathcal{D}$ a morphism of profunctors is a functor $F : \mathcal{P} \rightarrow \mathcal{P}'$ which restricts to the identity on $\mathcal{C}$ and $\mathcal{D}$.³ Let $\mathbf{Prof}(\mathcal{C}, \mathcal{D})$ denote the resulting category.

Remark 3.2. A profunctor $\mathcal{P} : \mathcal{C} \nrightarrow \mathcal{D}$ can equivalently be defined as a functor $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$, see [Sch+17, Example 2.19] and [Joy22]. Moreover, the Cartesian closure of $\mathbf{Cat}$ provides an equivalence $\mathbf{Prof}(\mathcal{C}, \mathcal{D}) \simeq [\mathcal{C}^{\text{op}} \times \mathcal{D}, \mathbf{Set}] \simeq [\mathcal{C}^{\text{op}}, \mathbf{Set}^{\mathcal{D}}]$ which allows us to think of a profunctor $\mathcal{P}$ also as a functor $\mathcal{P} : \mathcal{C}^{\text{op}} \rightarrow \mathbf{Set}^{\mathcal{D}}$. Each of these ways of thinking about profunctors suggests a different syntactic presentation.

Definition 3.3. Given profunctors $\mathcal{P} : \mathcal{C} \nrightarrow \mathcal{D}$ and $\mathcal{Q} : \mathcal{D} \nrightarrow \mathcal{E}$, let $\mathcal{P} \odot \mathcal{Q} : \mathcal{C} \nrightarrow \mathcal{E}$ denote the profunctor defined objectwise by the coend

$$(\mathcal{P} \odot \mathcal{Q})(c, e) = \int^{d \in \mathcal{D}} \mathcal{P}(c, d) \times \mathcal{Q}(d, e). \quad (1)$$

We call this the **composite profunctor** of $\mathcal{P}$ and $\mathcal{Q}$.

²Also see [Sch+17, Remark 4.21] for a similar construction for algebraic theories.

³There is a more general notion of morphism between possibly non-parallel profunctors, obtained from identifying profunctors with the category $\mathbf{Cat}/\mathbf{2}$ of small categories over $\mathbf{2}$. We postpone discussion of this notion of morphism to Section 5.

We now give a concrete description of the sets $(\mathcal{P} \odot \mathcal{Q})(c, e)$, for $c \in \mathcal{C}$ and $e \in \mathcal{E}$, which we will take as our concrete definition of $(\mathcal{P} \odot \mathcal{Q})$. Let $\simeq_0$ denote the relation on $\sum_{d \in \mathcal{D}} \mathcal{P}(c, d) \times \mathcal{Q}(d, e)$ defined by $(p, \mathcal{Q}(g, 1)(q')) \simeq_0 (\mathcal{P}(1, g)(p), q')$ for all morphisms $g : d \rightarrow d'$ in $\mathcal{D}$, $p \in \mathcal{P}(c, d)$, and $q' \in \mathcal{Q}(d', e)$.

Let $\simeq$ denote the smallest equivalence relation containing $\simeq_0$. Then we set

$$(\mathcal{P} \odot \mathcal{Q})(c, e) = \left(\sum_{d \in \mathcal{D}} \mathcal{P}(c, d) \times \mathcal{Q}(d, e) \right) / \simeq. \quad (2)$$

We let $\langle p, q \rangle$ denote the equivalence class of the pair (p, q) with $p \in \mathcal{P}(c, d)$ and $q \in \mathcal{Q}(d, e)$. If $f : c' \rightarrow c$ and $h : e \rightarrow e'$ are morphisms in $\mathcal{C}$ and $\mathcal{E}$ respectively, then $(\mathcal{P} \odot \mathcal{Q})(f, h)(\langle p, q \rangle) = \langle \mathcal{P}(f, 1)(p), \mathcal{Q}(1, h)(q) \rangle$.

It is not hard to show that $\odot$ defines a weakly associative and weakly unital⁴ composition operation. This allows us to think of profunctors as the horizontal arrows of a double category. We will explore this idea more fully in Section 5.

3.1 Uncurried Profunctor Presentations

Definition 3.4. Given category signatures Σ_0 and Σ_1 , an **uncurried profunctor signature** Π from Σ_0 to Σ_1 consists of a set $\text{Fun}(\Pi)$ whose elements we call **profunctor function symbols**, along with two functions $s : \text{Fun}(\Pi) \rightarrow \text{Sort}(\Sigma_0)$, $t : \text{Fun}(\Pi) \rightarrow \text{Sort}(\Sigma_1)$. For $x \in \text{Fun}(\Pi)$, we write $x : c \rightarrow d$ to mean that $s(x) = c$ and $t(x) = d$. We define the **category signature** $|\Pi|$ **associated to** Π by setting $\text{Sort}(|\Pi|) := \text{Sort}(\Sigma_0) + \text{Sort}(\Sigma_1)$, and $\text{Fun}(|\Pi|) := \text{Fun}(\Sigma_0) + \text{Fun}(\Pi) + \text{Fun}(\Sigma_1)$.

Given a path $p : c \rightarrow d$ in $|\Pi|$, we say p is a **cross-path** if $c \in \text{Sort}(\Sigma_0)$ and $d \in \text{Sort}(\Sigma_1)$. Let $\text{CPath}(\Pi)$ denote the set of cross-paths of Π . By an **uncurried profunctor equation** on Π , we mean an equation $p = q : c \rightarrow d$ between cross-paths $p : c \rightarrow d$ and $q : c \rightarrow d$ of Π . Given category presentations C and D , a (C, D) -**uncurried profunctor presentation** P is a pair $P = (P_\Pi, P_E)$ consisting of an uncurried profunctor signature P_Π from C_Σ to D_Σ and a set P_E of uncurried profunctor equations over P_Π . Let $\text{Fun}(P) := \text{Fun}(P_\Pi)$, and $\text{CPath}(P) := \text{CPath}(P_\Pi)$. We define the **associated category presentation** $|P|$ by $|P|_\Sigma := |P_\Pi|$ and $|P|_E := C_E + P_E + D_E$. We write $p =_P q$ to mean that $p = q$ is an equation in P_E . We define the relation $\approx_P$ as the restriction of $\approx_{|P|}$ to $\text{CPath}(P)$. (See Lemma B.1 for an inductive characterization of $\approx_P$.)

Definition 3.5. Given uncurried profunctor presentations $P, P' : C \rightarrow D$, a **morphism of uncurried profunctor presentations** $F : P \rightarrow P'$ is a function $F : \text{Fun}(P_\Pi) \rightarrow \text{CPath}(P'_\Pi)$ such that if $p : c \rightarrow d$, then $F(p) : c \rightarrow d$, and such that $|F| : |P| \rightarrow |P'|$ is a morphism of category presentations, where $|F| : |P| \rightarrow |P'|$ is the extension of F to $|P|$ acting as the identity on C and D . We extend F to cross-paths by $F(p) := |F|(p)$ (using Definition 2.4). We say that two such morphisms F and F' are **provably equal**, and write $F \approx F'$, if $|F| \approx |F'|$.

Let $\mathbf{UnCurr}(C, D)$ denote the category of uncurried profunctor presentations from C to D , where the composition $G \circ F$ of morphisms $F : P \rightarrow P'$ and $G : P' \rightarrow P''$ is defined as the function $\text{Fun}(P_\Pi) \rightarrow \text{CPath}(P''_\Pi)$ given by $p \mapsto |G|(F(p))$. Let $\mathbf{FinUnCurr}(C, D)$ denote the corresponding full subcategory of finite uncurried profunctor presentations, and let $\mathbf{UnCurr}_\approx(C, D) := \mathbf{UnCurr}(C, D) / \approx$.

Definition 3.6. Given an uncurried profunctor presentation $P : C \rightarrow D$, we let $\langle P \rangle$ denote the profunctor $\langle P \rangle$, with $\pi : \langle P \rangle \rightarrow \mathbf{2}$ sending all sorts of C to 0 and all sorts of D to 1. This construction extends to a functor $\langle - \rangle : \mathbf{UnCurr}(C, D) \rightarrow \mathbf{Prof}(C, D)$ for all C and D .

⁴The unit profunctor on a category $\mathcal{C}$ is given by the Hom functor $\text{Hom}_{\mathcal{C}} : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set}$.

The following result can be proved using a similar argument as in the proof of Proposition 2.12.

Proposition 3.7. For all category presentations C and D , the induced functor $\langle - \rangle : \mathbf{UnCurr}_{\approx}(C, D) \rightarrow \mathbf{Prof}(\langle C \rangle, \langle D \rangle)$ is an equivalence.

Proof sketch. As in the proof sketch for Proposition 2.12, we define a quasi-inverse $\text{Can} : \mathbf{Prof}(\langle C \rangle, \langle D \rangle) \rightarrow \mathbf{UnCurr}_{\approx}(C, D)$ by assigning to each profunctor $\mathcal{P}$ a canonical uncurried presentation between C and D whose set of profunctor symbols is the set of cross-morphisms of $\mathcal{P}$ and whose profunctor equations consist of pairs of paths in $\text{Can}(\mathcal{P})$ whose compositions in $\mathcal{P}$ are equal. $\square$

Definition 3.8. Given category presentations C and D , we say that a profunctor $\mathcal{P} : \langle C \rangle \rightarrow \langle D \rangle$ is **finitely uncurried presentable** if there exists a finite uncurried profunctor presentation $P : C \rightarrow D$ and an isomorphism $\mathcal{P} \cong \langle P \rangle$ in $\mathbf{Prof}(\langle C \rangle, \langle D \rangle)$.

Uncurried profunctor presentations are natural from the point of view of profunctors as categories over **2**. However, if we restrict to *finite* presentations, then they are not suitable for presenting profunctor composition, as the following result illustrates.

Proposition 3.9. The class of finitely uncurried presentable profunctors is not closed under composition.

Proof. Consider monoid presentations $C = \langle \emptyset \mid \emptyset \rangle$, $D = \langle f \mid \emptyset \rangle$ and $E = \langle \emptyset \mid \emptyset \rangle$. For clarity, we refer to their unique sorts by c, d and e , respectively. Let $P : C \rightarrow D$ be the uncurried profunctor presentation with one profunctor function symbol $p : c \rightarrow d$ and no equations, and let $Q : D \rightarrow E$ be the uncurried profunctor presentation with one profunctor function symbol $q : d \rightarrow e$ and no equations. Then $(\langle P \rangle \odot \langle Q \rangle)(c, e) = \{[p.f^n.q] \mid n \geq 0\}$ is an infinite set. Now suppose that $R : C \rightarrow E$ is an uncurried profunctor presentation such that $\langle R \rangle \cong \langle P \rangle \odot \langle Q \rangle$. Then in particular $\langle R \rangle(c, e) \cong (\langle P \rangle \odot \langle Q \rangle)(c, e)$ is infinite. But C and E have no function symbols which implies that R must have infinitely many profunctor function symbols. $\square$

Proposition 3.9 poses a potential problem for applications of computational categorical algebra. For instance, this is a problem in categorical database theory, where profunctor presentations are used to represent database queries [SW17]. Curried profunctor presentations, which we introduce later in this section, do not share this deficiency. In order to define them, though, we first introduce instance presentations.

3.2 Instance Presentations

Notation 3.10. Given a category $\mathcal{C}$, we will refer to a functor $\mathcal{I} : \mathcal{C} \rightarrow \mathbf{Set}$ as an **instance** on $\mathcal{C}$, or **$\mathcal{C}$ -instance**. We also write $\mathcal{C}\text{-Inst} := \mathbf{Set}^{\mathcal{C}}$.

An instance $\mathcal{C} \rightarrow \mathbf{Set}$ is equivalently a functor $\mathcal{I} : \mathbf{1}^{\text{op}} \times \mathcal{C} \rightarrow \mathbf{Set}$, i.e. a profunctor $\mathcal{I} : \mathbf{1} \rightarrow \mathcal{C}$. This allows us to define instance presentations as a particular case of uncurried profunctor presentations.

Definition 3.11. Let $\mathbf{1}$ be the category presentation with a single sort $*$ and no function symbols or equations. Given a category presentation C , a **C -instance signature** is an uncurried profunctor signature from $\mathbf{1}$ to C , and a **C -instance presentation** is an uncurried profunctor presentation from $\mathbf{1}$ to C . Let $\mathcal{C}\text{-InstPr} := \mathbf{UnCurr}(\mathbf{1}, C)$ and $\mathcal{C}\text{-InstPr}_{\approx} := \mathbf{UnCurr}_{\approx}(\mathbf{1}, C)$.

Notice that the semantics of a C -instance presentation I is a functor $\langle I \rangle : \langle \mathbf{1} \rangle^{\text{op}} \times \langle C \rangle \rightarrow \mathbf{Set}$. Since $\langle \mathbf{1} \rangle^{\text{op}} \cong \mathbf{1}$, it is equivalently a $\langle C \rangle$ -instance $\langle I \rangle : \langle C \rangle \rightarrow \mathbf{Set}$.

Notation 3.12. Given a C -instance presentation I , we call the profunctor function symbols of I **generators**, writing $\text{Gen}(I) := \text{Fun}(I)$, and we call cross-paths of I **terms**, writing $\text{Term}(I) := \text{CPath}(I)$. Note that $|I|_E = I_E + C_E$. Since there is only one sort in 1 , we abbreviate $(x : * \rightarrow c) \in \text{Gen}(I)$ to $(x : c) \in \text{Gen}(I)$, to be read as “ x is a term in I of type c ”. As in Definition 3.5, F also acts on terms by $F(t) := |F|(t)$.

Example 3.13. Given a monoid presentation M , we refer to a M -instance presentation I as a **M -action presentation**, and we write $I = \langle \text{Gen}(I) \mid I_E \rangle_M$. For example, let $N = \langle s \mid \emptyset \rangle$ be a presentation for the free monoid on one generator, i.e. $\langle N \rangle \simeq \mathbb{N}$. Then $I = \langle x, y \mid x.s = x \rangle_N$ presents the set $\langle I \rangle(*) = \{[x], [y], [y.s], [y.s.s], \dots\}$ together with the $\langle N \rangle$ -action $\langle I \rangle(s)([y.s^k]) = [y.s^{k+1}]$ and $[x] \cdot s := [x]$. This can be identified with the set $\mathbb{N} \sqcup \{\infty\}$ and the $\mathbb{N}$ -action $n \cdot s := n + 1$ and $\infty \cdot s := \infty$.

Remark 3.14. A profunctor $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ can be equivalently be thought of as a functor $\mathcal{P} : \mathcal{C}^{\text{op}} \times \mathcal{D} \rightarrow \mathbf{Set}$. This could be thought to give rise to a different notion of presentation: indeed, it is possible to define an involution $C \mapsto C^{\text{op}}$ and an operation $(C, D) \mapsto C \times D$ in $\mathbf{CatPr}$ such that one may define a profunctor presentation as a $(C^{\text{op}} \times D)$ -instance presentation. However, $(C^{\text{op}} \times D)$ -instance presentations turn out to be equivalent to (C, D) -uncurried profunctor presentations.

In contrast, if we use the Cartesian closure of $\mathbf{Cat}$ to think of a profunctor $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ as a functor $\mathcal{P} : \mathcal{C}^{\text{op}} \rightarrow \mathcal{D}\text{-Inst}$, it becomes intuitive to define a profunctor presentation $P : C \rightarrow D$ as a diagram of D -instance presentations, i.e. a family of D -instance presentations and morphisms between them indexed by the sorts and function symbols of C . We will show that in this case the resulting notion turns out to have starkly different properties. This might be surprising since the equivalence of categories $\mathbf{Prof}(\mathcal{C}, \mathcal{D}) \simeq [C^{\text{op}}, \mathcal{D}\text{-Inst}]$ is widely used throughout the categorical literature, often without mention.

3.3 Curried Profunctor Presentations

Definition 3.15. Given category presentations C and D , a **curried profunctor presentation** $P : C \rightarrow D$ consists of an assignment of a D -instance presentation $P(c)$ to every sort $c \in C$, and an assignment of a D -instance presentation morphism $P(f) : P(c') \rightarrow P(c)$ to every function symbol $(f : c \rightarrow c') \in \text{Fun}(C)$, such that for each equation $p =_C p'$ we have $P(p) \approx P(p')$, where $P(p)$ is defined as $P(p) := P(p_n) \circ \dots \circ P(p_1)$ for $p \equiv p_1 \cdot \dots \cdot p_n$. We say $P : C \rightarrow D$ is **finite** if C and D are finite and $P(c)$ is finite for all c .

Example 3.16. Given monoid presentations M and N , we refer to an (un)curried profunctor presentation $P : M \rightarrow N$ as a **(un)curried (M, N) -bimodule presentation**. For example, consider the previously introduced monoid presentations $M = \langle f, g \mid f.g = g.f \rangle$ and $N = \langle s \mid \emptyset \rangle$. Then a curried (M, N) -bimodule presentation is given by a choice of an N -action presentation $P(*)$ together with endomorphisms $P(f), P(g) : P(*) \rightarrow P(*)$ such that $P(g) \circ P(f) \approx P(f) \circ P(g)$. Let $P(*)$ be the N -action presentation of Example 3.13, $P(*) := \langle x, y \mid x.s = x \rangle_N$. As for the endomorphisms, not only must they commute (up to provable equality), but they must also respect the fact that x is a fixed point (up to provable equality) of the action. This means that they must both send x to a term provably equal to x . The reader may check that a valid choice is as follows: $P(f)(x) := P(g)(x) := x$, $P(f)(y) := y.s$ and $P(g)(y) := y.s.s$.

Since curried profunctor presentations are essentially diagrams of instance presentations, morphisms between them can be thought of as presentations of natural transformations.

Definition 3.17. Given curried profunctor presentations $P, P' : C \rightarrow D$, a **morphism of curried profunctor presentations** $F : P \rightarrow P'$ consists of a collection of D -instance presentation morphisms $F_c : P(c) \rightarrow P'(c)$, indexed by the sorts c of C , such that for each function symbol $f : c \rightarrow c'$ in C , the following

diagram commutes up to provable equality:

$$\begin{array}{ccc}
 P(c') & \xrightarrow{F_{c'}} & P'(c') \\
 P(f) \downarrow & \approx & \downarrow P'(f) \\
 P(c) & \xrightarrow{F_c} & P'(c)
 \end{array} \tag{3}$$

It follows that (3) still commutes when f is a path, by induction on f ; thus these morphisms can be composed in the natural way. Let $\mathbf{Curr}(C, D)$ denote the category of curried profunctor presentations from C to D , and if C and D are finite, let $\mathbf{FinCurr}(C, D)$ denote the full subcategory of finite curried profunctor presentations.

Given $P, P' : C \rightarrowtail D$, we say that two curried profunctor presentation morphisms $F, G : P \rightarrow P'$ are **provably equal**, and write $F \approx G$, if $F_c \approx G_c$ for each $c \in C$.

Definition 3.18. Given a curried profunctor presentation $P : C \rightarrowtail D$, let $\langle P \rangle$ denote the functor $\langle P \rangle : \langle C \rangle^{\text{op}} \rightarrow \langle D \rangle$ -**Inst** given by $\langle P \rangle(c) := \langle P(c) \rangle$ for each $c \in \langle C \rangle$ and $\langle P \rangle([f]) := \langle P(f_n) \rangle \circ \dots \circ \langle P(f_1) \rangle$ for each morphism $[f] = [f_1 \cdot \dots \cdot f_n]$ in $\langle C \rangle$. Notice that well definition follows from the definition of curried profunctor presentation. Moreover, $\langle P \rangle$ thus defined is equivalently a profunctor $\langle P \rangle : \langle C \rangle^{\text{op}} \times \langle D \rangle \rightarrow \mathbf{Set}$ with $\langle P \rangle(c, d) = \langle P(c) \rangle(d)$. It is easy to see that this extends to a functor $\langle - \rangle : \mathbf{Curr}(C, D) \rightarrow \mathbf{Prof}(\langle C \rangle, \langle D \rangle)$.

We say a profunctor $\mathcal{P} : \langle C \rangle \rightarrowtail \langle D \rangle$ is **finitely curried presentable** if there exists a finite curried profunctor presentation $P : C \rightarrowtail D$ such that $\langle P \rangle \cong \mathcal{P}$.

Remark 3.19. A curried profunctor presentation $P : C \rightarrowtail D$ does not determine a functor $\langle C \rangle^{\text{op}} \rightarrow D$ -**InstPr**. That would require the diagram (3) to commute strictly, rather than up to provable equality. Now P does determine a functor $P : \langle C \rangle^{\text{op}} \rightarrow D$ -**InstPr**_≈, but the converse is not true, because the morphisms $P(f)$ are actual data, rather than equivalence classes of such data.

Given a category $\mathcal{C}$ and a congruence $\sim$ on $\mathcal{C}$, we can also consider the pair $(\mathcal{C}, \sim)$ as a bicategory where given morphisms f, g in $\mathcal{C}$, there is a unique 2-morphism $f \Rightarrow g$ if and only if $f \sim g$. With this definition P defines a pseudofunctor $\langle C \rangle^{\text{op}} \rightarrow (D\text{-}\mathbf{InstPr}, \approx)$. We leave the development of the consequences of this observation to future work.

It is not hard to show that every profunctor can be presented by an uncurried profunctor presentation or a curried profunctor presentation, using ideas similar to the proof of Proposition 2.12. However, if we restrict to *finite* uncurried and curried profunctor presentations, then the situation is very different, as the following result shows.

Lemma 3.20. There exist profunctors that are finitely uncurried presentable but not finitely curried presentable.

Proof. As in the proof of Prop. 3.9, let $D = \langle f \mid \emptyset \rangle$ and $E = \langle \emptyset \mid \emptyset \rangle$ denote monoid presentations with unique sorts d and e respectively, and let $Q : D \rightarrowtail E$ be the uncurried bimodule presentation given by $\text{Fun}(Q) = \{q : d \rightarrow e\}$ and no equations. Then $\langle Q \rangle(d, e) = \{[p], [f \cdot p], [f \cdot f \cdot p], \dots\}$ is an infinite set, and $\langle Q \rangle(f)([f^n \cdot p]) = [f^{n+1} \cdot p]$. If $Q' : D \rightarrowtail E$ is a curried bimodule presentation for $\langle Q \rangle$, then $Q'(d)$ must be a E -instance presentation such that $\langle Q'(d) \rangle(e) \cong \langle Q \rangle(d, e)$. This implies that $Q'(d)$ has infinitely many terms. Since E has no function symbols, $Q'(d)$ must have infinitely many generators. Thus it cannot be a finite curried bimodule presentation. $\square$

In contrast to Proposition 3.9, we define a notion of syntactic composition for curried profunctor presentations that is semantically correct.

Definition 3.21. Given category presentations C , D , and E , curried profunctor presentations $P : C \rightarrowtail D$ and $Q : D \rightarrowtail E$, and a sort $c \in C$, we define the E -instance presentation $(P \circledast Q)(c)$ as follows. let $(P \circledast Q)(c)_\Pi$ denote the E -instance signature whose set of generators consists of pairs of generators $(p : d) \in \text{Gen}(P(c))$ and $(q : e) \in \text{Gen}(Q(d))$, which we write as $(p \otimes q : e)$. We now extend the operation $\otimes$ to accept a pair (s, t) of terms with $(s : d) \in \text{Term}(P(c))$, $(t : e) \in \text{Term}(Q(d))$ and return a term in $(P \circledast Q)(c)$. Firstly, if $(p : d) \in \text{Gen}(P(c))$ and $(t \equiv q.h : e) \in \text{Term}(Q(d))$, where q is a generator of $Q(d)$ and h is a path in E , then set $p \otimes t \equiv (p \otimes q).h$. Secondly, if $(s \equiv p.g : d) \in \text{Term}(P(c))$ and $(t : e) \in \text{Term}(Q(d))$, then set $s \otimes t \equiv p \otimes Q(g)(t)$.

Let $(P \circledast Q)(c)$ be the E -instance presentation with signature $(P \circledast Q)(c)_\Pi$ and set of equations $(P \circledast Q)(c)_E$ consisting of equations of the form

- $(s \otimes q =_{(P \circledast Q)(c)} s' \otimes q)$, where $(s =_{P(c)} s' : d)$ and $(q : e) \in \text{Gen}(P(d))$, or
- $(p \otimes t =_{(P \circledast Q)(c)} p \otimes t')$, where $(p : d) \in \text{Gen}(P(c))$ and $(t =_{Q(d)} t' : e)$.

Lemma 3.22. Given curried profunctor presentations $P : C \rightarrowtail D$ and $Q : D \rightarrowtail E$, the operation $\otimes$ respects provable equality. In other words, if $(s \approx_{P(c)} s' : d)$ and $(t \approx_{Q(d)} t' : e)$, then $(s \otimes t \approx_{(P \circledast Q)(c)} s' \otimes t' : e)$.

Definition 3.23. Given curried profunctor presentations $P : C \rightarrowtail D$ and $Q : D \rightarrowtail E$ as above, we define the **composite presentation** $P \circledast Q$ as follows. We have already defined an E -instance $(P \circledast Q)(c)$ for each $c \in \text{Sort}(C)$ in Definition 3.21. Given a function symbol $f : c \rightarrow c'$ in C , we define a function $(P \circledast Q)(f) : \text{Gen}((P \circledast Q)(c')) \rightarrow \text{Term}((P \circledast Q)(c))$ by setting $(P \circledast Q)(f)(p \otimes q) \equiv P(f)(p) \otimes q$.

Lemma 3.24. Given a C -function symbol $f : c \rightarrow c'$ in C , the function $(P \circledast Q)(f)$ is a morphism of E -instances. Furthermore $P \circledast Q$ is a well-defined curried profunctor presentation from C to E .

Lemma 3.25. The assignment $(P, Q) \mapsto P \circledast Q$ defines a functor

$$\circledast : \mathbf{Curr}(C, D) \times \mathbf{Curr}(D, E) \rightarrow \mathbf{Curr}(C, E).$$

Now that we have seen that the composition operation $\circledast$ is well defined, we now show that its semantics are given by $\odot$. From the algorithmic point of view, this provides a proof of correctness of the composition algorithm given implicitly by $\circledast$.

Theorem 3.26. Given curried profunctor presentations $P : C \rightarrowtail D$ and $Q : D \rightarrowtail E$, there is an isomorphism

$$\mu^{P, Q} : \langle P \rangle \odot \langle Q \rangle \rightarrow \langle P \circledast Q \rangle$$

of profunctors, natural in P and Q . Furthermore, if $P : C \rightarrowtail D$ and $Q : D \rightarrowtail E$ are finite curried profunctor presentations, then $P \circledast Q$ is a finite curried profunctor presentation. In particular, composition of profunctors preserves finite curried presentability.

Notice that we can extract an algorithm for computing the composition of two finite curried profunctor presentations by inspection of Definition 3.23. In this sense, Theorem 3.26 proves the correctness of this algorithm with respect to the semantics.

Remark 3.27. Theorem 3.26 gives an alternative proof of Lemma 3.20. Indeed, consider profunctors $\langle P \rangle$ and $\langle Q \rangle$ in Proposition 3.9. It is not hard to see that $\langle P \rangle$ is finitely curried presentable and $\langle P \rangle \odot \langle Q \rangle$ is not finitely curried presentable. Hence, by Theorem 3.26 $\langle Q \rangle$ is not finitely curried presentable.

Example 3.28. Here we give an example of curried profunctor presentation composition. Recall the (M, N) -bimodule presentation $P : M \rightarrowtail N$ from Example 3.16. Now let O be the monoid presentation $O = \langle t \mid \emptyset \rangle$ and let $Q : N \rightarrowtail O$ be the (N, O) -bimodule presentation $Q(*) = \langle q \mid q.t.t = q.t \rangle_O$ with $Q(s)(q) \equiv q.t$. Then the composite presentation $(P \circledast Q) : M \rightarrowtail O$ is given by

$$(P \circledast Q)(*) = \langle x \otimes q, y \otimes q \mid (x \otimes q).t.t = (x \otimes q).t, (y \otimes q).t.t = (y \otimes q).t, (x \otimes q).t = x \otimes q \rangle_O$$

with

- $(P \circledast Q)(f)(x \otimes q) \equiv P(f)(x) \otimes q \equiv x \otimes q$,
- $(P \circledast Q)(f)(y \otimes q) \equiv P(f)(y) \otimes q \equiv y.s \otimes q \equiv y \otimes Q(s)(q) \equiv (y \otimes q).t$,
- $(P \circledast Q)(g)(x \otimes q) \equiv P(g)(x) \otimes q \equiv x \otimes q$, and
- $(P \circledast Q)(g)(y \otimes q) \equiv P(g)(y) \otimes q \equiv y.s.s \otimes q \equiv y \otimes Q(s.s)(q) \equiv (y \otimes q).t.t$.

In turn, $P \circledast Q$ presents the profunctor given by

$$\langle P \circledast Q \rangle(*, *) = \{[x \otimes q], [y \otimes q], [(y \otimes q).t]\}$$

$$\langle P \circledast Q \rangle([f], 1) = \langle P \circledast Q \rangle([g], 1) : [x \otimes q] \mapsto [x \otimes q], \quad [y \otimes q] \mapsto [y \otimes q.t], \quad [y \otimes q.t] \mapsto [y \otimes q.t]$$

$$\langle P \circledast Q \rangle(1, [t]) : [x \otimes q] \mapsto [x \otimes q], \quad [y \otimes q] \mapsto [y \otimes q.t], \quad [y \otimes q.t] \mapsto [y \otimes q.t]$$

Notice that $P \circledast Q$ is not a minimal presentation for $\langle P \circledast Q \rangle$, e.g. the equation $(x \otimes q).t.t = (x \otimes q).t$ is redundant.

4 Curryable Profunctor Presentations

As we have seen, finite curried profunctor presentations, unlike uncurried presentations, admit composition. However, the definition of $\mathbf{Curr}(C, D)$ is more involved than that of $\mathbf{UnCurr}(C, D)$. It is therefore desirable to find some subclass of uncurried profunctor presentations which behaves like the class of curried presentations.

Definition 4.1. Given a curried profunctor presentation $P : C \rightarrowtail D$, let $\bar{P}$ denote the uncurried profunctor presentation defined as follows. Let $\text{Fun}(\bar{P}) := \{\bar{p} \mid c \in C, p \in \text{Gen}(P(c))\}$, with typing $(\bar{p} : c \rightarrow d)$ for $(p : d) \in \text{Gen}(P(c))$. This induces a function $\overline{(-)} : \text{Term}(P(c)) \rightarrow \text{CPath}(\bar{P})$ for every $c \in C$, by setting $\bar{p}.g \equiv \bar{p}.g$ for every generator p and D -path g . Let $\bar{P}_E := \{\bar{t} = \bar{t}' \mid c \in C, t =_{P(c)} t'\} + \{f.\bar{p} = \bar{P}(f)(\bar{p}) \mid (f : c \rightarrow c') \in \text{Fun}(C), p \in \text{Gen}(P(c'))\}$. Given a morphism $F : P \rightarrow Q$ of curried profunctor presentations, we define $\bar{F} : \bar{P} \rightarrow \bar{Q}$ by setting $\bar{F}(\bar{p}) := \bar{F}_c(p)$.

Proposition 4.2. Given category presentations C and D , the construction given in Definition 4.1 defines a functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ which preserves provable equality of morphisms and makes the following diagram commute up to isomorphism

$$\begin{array}{ccc} \mathbf{Curr}(C, D) & \xrightarrow{\overline{(-)}} & \mathbf{UnCurr}(C, D) \\ & \searrow \langle (-) \rangle & \swarrow \langle (-) \rangle \\ & \mathbf{Prof}(\langle C \rangle, \langle D \rangle) & \end{array}$$

Furthermore, $\overline{(-)}$ restricts to a functor $\overline{(-)} : \mathbf{FinCurr}(C, D) \rightarrow \mathbf{FinUnCurr}(C, D)$.

Remark 4.3. By Definition 4.1 and Lemma 3.20, we conclude that the class of finitely curried presentable profunctors is a proper subclass of the class of finitely uncurried presentable profunctors.

We can begin to address our current goal by characterizing the image of the functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$.

Definition 4.4. Let $P : C \rightarrowtail D$ be an uncurried profunctor presentation. By a **short left cross-path** for short, we mean a path in P of the form $f.p : c \rightarrow d$, where $f : c \rightarrow c'$ is a function symbol in C and $(p : c' \rightarrow d) \in \text{Fun}(P)$. By a **right cross-path**, we mean a path in P of the form $p.g : c \rightarrow d'$ where $(p : c \rightarrow d) \in \text{Fun}(P)$ and $g : d \rightarrow d'$ is a path in D .

Definition 4.5. Given an uncurried profunctor presentation $P : C \nrightarrow D$, and $c \in C$, let P^c denote the D -instance presentation with $\text{Gen}(P^c) = \{(p : d) \mid p : c \rightarrow d \in \text{Fun}(P)\}$ and $P_E^c = \{(r = r') \in P_E \mid s(r) = c\}$.

Definition 4.6. Given an uncurried profunctor presentation $P : C \nrightarrow D$, we say that P is **nongenerative** if for every short left cross-path l in P , there exists a right cross-path r such that $l \approx_P r$. We say that P is **conservative**⁵ if whenever t and t' are right cross-paths such that $t \approx_P t'$ we also have $t \approx_{P^c} t'$.

Lemma 4.7. Let $P : C \nrightarrow D$ be a nongenerative uncurried profunctor presentation. Then for every cross-path t of P there exists a right cross-path r such that $t \approx_P r$.

Definition 4.8. Given profunctors $\mathcal{P}, \mathcal{P}' : C \nrightarrow \mathcal{D}$ and a morphism $F : \mathcal{P} \rightarrow \mathcal{P}'$, we say that F is **full on cross-morphisms** if for all $c \in C$ and $d \in \mathcal{D}$, the function $F_{(c,d)} : \mathcal{P}(c, d) \rightarrow \mathcal{P}'(Fc, Fd)$ is surjective.

Lemma 4.9. Given an uncurried profunctor presentation $P : C \nrightarrow D$, for every $c \in C$, there is an obvious inclusion functor⁶ $\iota^c : \langle P^c \rangle \rightarrow \langle P \rangle$. Thus P is nongenerative if and only if for every $c \in C$, ι^c is full on cross-morphisms. Similarly P is conservative if and only if for every $c \in C$, ι^c is faithful.

Definition 4.10. We call an uncurried presentation that is both nongenerative and conservative a **curryable** profunctor presentation. We say that a morphism $F : P \rightarrow P'$ of uncurried profunctor presentations is **rightward** if it sends profunctor function symbols $p \in \text{Fun}(P)$ to right cross-paths in P' . Given category presentations C and D , let $\mathbf{Crble}(C, D)$ denote the subcategory of $\mathbf{UnCurr}(C, D)$ whose objects are curryable profunctor presentations and whose morphisms are rightward morphisms. Let $\mathbf{FinCrble}(C, D)$ denote the full subcategory on the finite curryable profunctor presentations.

Our particular definition of $\mathbf{Crble}(C, D)$ is motivated by the following result.

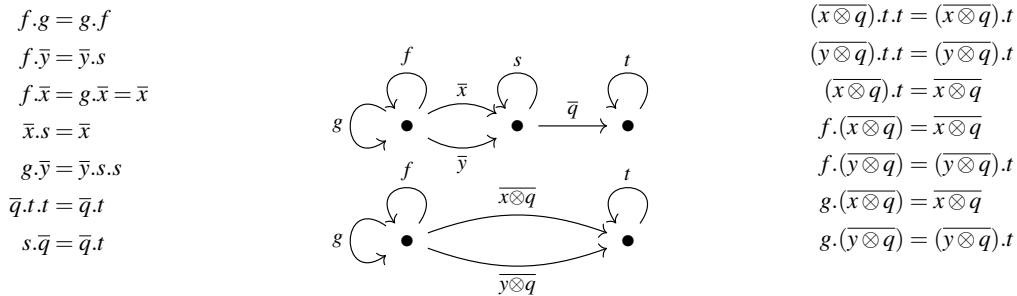
Theorem 4.11. Given category presentations C and D , the functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ takes values in curryable profunctor presentations and rightward morphisms. Furthermore the corestrictions $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Crble}(C, D)$ and $\overline{(-)} : \mathbf{FinCurr}(C, D) \rightarrow \mathbf{FinCrble}(C, D)$ are equivalences of categories which preserve and reflect provable equality.

Corollary 4.12. The class of profunctors which admit finite curryable profunctor presentations is closed under composition.

Conservativity is a necessary condition for Corollary 4.12 to hold, as the following result shows.

Proposition 4.13. There exist finite nongenerative uncurried profunctor presentations $P : C \nrightarrow D$ and $Q : D \nrightarrow E$ between finite category presentations such that $\langle P \rangle \odot \langle Q \rangle$ does not admit a finite uncurried presentation.

Remark 4.14. We can revisit Example 3.28 replacing curried profunctor presentations P and Q for their uncurried counterparts $\overline{P}$ and $\overline{Q}$. We can then visualize the overall situation as follows:



⁵The name comes from the fact that a conservative presentation $P : C \nrightarrow D$ is a conservative extension of each of the P^c , when we consider P and P^c as algebraic theories. See [SW17, Section 4.1].

⁶Where here we think of $\langle P^c \rangle$ as a profunctor $\langle P^c \rangle : * \nrightarrow D$.

The diagram on top, together with the equations on the left, represents the situation before computing the composite, i.e. $\overline{P}$ and $\overline{Q}$ drawn together. Meanwhile, the bottom diagram together with the equations on the right represents $\overline{P \otimes Q}$. This perspective on composition of profunctor presentations, using curryable presentations, has the benefit of being more easily visualized.

Remark 4.15. We have chosen to present a result on a category of curryable profunctor presentations from C to D which is *equivalent* to $\mathbf{Curr}(C, D)$. However, other options are possible, depending on the motivation.

From a theoretical perspective, we may find the restriction to rightward morphisms undesirable, since it makes $\mathbf{Crble}(C, D)$ a non-full subcategory of $\mathbf{Curr}(C, D)$. We can solve this by relaxing our notion of equivalence. If we now consider $\mathbf{Crble}(C, D)$ to be the full subcategory of $\mathbf{Uncurr}(C, D)$ spanned by curryable presentations, then $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Crble}(C, D)$ is no longer an equivalence, but is a *biequivalence* between bicategories (considering provable equalities as 2-morphisms as in Remark 3.19) which restricts to a biequivalence between the corresponding full sub-bicategories of finite presentations.⁷

Alternatively, from the point of view of implementation we may be interested in being able to replace curried presentations by uncurried presentations through a bijective correspondence. In this case we could then strengthen nongenerativity to require that all equations are either between right cross-paths or between a short left cross-path and a right cross-path, and every short left cross-path appears in some equation exactly once. Using this notion of nongenerativity, the functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Crble}(C, D)$ actually becomes an isomorphism of categories, which again restricts to an isomorphism between the corresponding full subcategories of finite presentations.

5 The Double Category of Curried Profunctor Presentations

A natural categorical structure to think about profunctors is that of double categories. By a double category, we mean what is sometimes called a weak or pseudo double category in the literature [Fer06; Fio07]. Namely we consider horizontal arrow composition to be weakly unital and associative, in contrast to strict double categories [JY20, Explanation 12.3.1]. The double category $\mathbb{P}\mathbf{prof}$ packages together the morphisms between profunctors and the composition operation of profunctors. Analogously, curried profunctor presentations can also be arranged in this way together with $\otimes$ into a double category $\mathbb{C}\mathbf{urr}$.

Theorem 5.1. There exists a double category $\mathbb{C}\mathbf{urr}$ such that the objects are category presentations and the horizontal morphisms are curried profunctor presentations.

The double category $\mathbb{C}\mathbf{urr}$ organizes the composition of curried profunctor presentations and their transformations in a way that mimics the structure of $\mathbb{P}\mathbf{prof}$. The isomorphism of Theorem 3.26 is now packaged into a double functor between these double categories.

Theorem 5.2. The construction $\langle - \rangle$ on curried profunctor presentations defines a double functor $\langle - \rangle : \mathbb{C}\mathbf{urr} \rightarrow \mathbb{P}\mathbf{prof}$.

Analogously to the case of $\mathbf{Curr}(C, D)$, we can quotient $\mathbb{C}\mathbf{urr}$ by an appropriate congruence of double categories induced by the provable equality relation.

⁷Our original functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Uncurr}(C, D)$ is already a biequivalence in this way, but it does not restrict to a biequivalence between the full sub-bicategories of finite presentations.

Proposition 5.3. The congruences $\approx$ on **Curr** and **CatPres** comprise a double congruence on **Curr**. Define $\mathbf{Curr}_{\approx}$ by quotienting **Curr** by $\approx$, as in Def. D.4. The semantics double functor $(|-|)$ induces a double equivalence $(|-|) : \mathbf{Curr}_{\approx} \rightarrow \mathbf{Prof}$.

We can also express the fact that composition of curried profunctor presentations preserves finiteness as the existence of a double subcategory $\mathbf{FinCurr} \subseteq \mathbf{Curr}$.

In future work we plan to study syntactic objects using double categories equipped with double congruences as extra structure in a way analogous to Remark 3.19, but in this case leading to a three-dimensional structure. We believe that this could constitute a general framework for reasoning about presentations of proarrows in double categories.

References

- [BBP99] Marek A Bednarczyk, Andrzej M Borzyszkowski, and Wiesław Pawłowski. “Generalized congruences-epimorphisms in Cat”. In: *Theory and Applications of Categories* 5.11 (1999), pp. 266–280 (cit. on p. 91).
- [Bor94] Francis Borceux. *Handbook of categorical algebra: volume 1, Basic category theory*. Vol. 1. Cambridge University Press, 1994. DOI: <https://doi.org/10.1017/CB09780511525858> (cit. on p. 91).
- [Fer06] Nelson Martins Ferreira. *Pseudo-categories*. 2006. DOI: <http://dx.doi.org/10.48550/arXiv.math/0604549>. arXiv: math/0604549 [math.CT] (cit. on pp. 100, 111).
- [Fio07] Thomas M. Fiore. *Pseudo Algebras and Pseudo Double Categories*. 2007. arXiv: math/0608760 [math.CT] (cit. on p. 100).
- [Gra19] Marco Grandis. *Higher dimensional categories: From double to multiple categories*. World Scientific, 2019. DOI: <https://doi.org/10.1142/11406> (cit. on p. 111).
- [Joy22] Andre Joyal. “Joyal’s CatLab distributors and barrels”. In: (2022). URL: <https://ncatlab.org/joyalcatlab/published/Distributors+and+barrels> (cit. on p. 92).
- [JY20] Niles Johnson and Donald Yau. *2-Dimensional Categories*. 2020. DOI: <https://doi.org/10.1093/oso/9780198871378.001.0001>. arXiv: 2002.06055 [math.CT] (cit. on pp. 100, 111).
- [Lor21] Fosco Loregian. *(Co) end calculus*. Vol. 468. Cambridge University Press, 2021. DOI: <https://doi.org/10.1017/9781108778657> (cit. on p. 112).
- [PLF22] Evan Patterson, Owen Lynch, and James Fairbanks. “Categorical Data Structures for Technical Computing”. In: *Compositionality* 4 (2022), p. 5. DOI: <https://doi.org/10.32408/compositionality-4-5> (cit. on p. 88).
- [Sch+17] Patrick Schultz, David Spivak, Christina Vasilakopoulou, and Ryan Wisnesky. “Algebraic Databases”. In: *Theory and Applications of Categories* 32.16 (2017), pp. 547–619 (cit. on pp. 88, 89, 92, 112).
- [Shu08] Michael Shulman. “Framed bicategories and monoidal fibrations.” In: *Theory and Applications of Categories* 20 (2008), pp. 650–738 (cit. on p. 114).
- [Spi12] David I Spivak. “Functorial data migration”. In: *Information and Computation* 217 (2012), pp. 31–51. DOI: <https://doi.org/10.1016/j.ic.2012.05.001> (cit. on p. 88).
- [Spi14] David I Spivak. *Category theory for the sciences*. MIT press, 2014 (cit. on p. 90).
- [SW15] David I Spivak and Ryan Wisnesky. “Relational foundations for functorial data migration”. In: *Proceedings of the 15th Symposium on Database Programming Languages*. 2015, pp. 21–28. DOI: <https://doi.org/10.1145/2815072.2815075> (cit. on p. 88).
- [SW17] Patrick Schultz and Ryan Wisnesky. “Algebraic data integration”. In: *Journal of Functional Programming* 27 (2017). DOI: <https://doi.org/10.1142/11406> (cit. on pp. 88, 89, 94, 99).

A Proofs for Section 2 (Category Presentations)

We repeat here, for ease of reference, the definition of provable equality.

Definition 2.3. Given a category presentation C , define a relation $\approx_C$ on paths of C inductively⁸ by the following inference rules:

1. For any equation $p =_C q$, $p \approx_C q$.
2. For any path p , $p \approx_C p$.
3. If $p \approx_C q$, then $q \approx_C p$.
4. If $p \approx_C q$ and $q \approx_C r$, then $p \approx_C r$.
5. For any C -function symbol $f : c' \rightarrow c''$, if $p \approx_C q : c \rightarrow c'$, then $p.f \approx_C q.f$.
6. For any C -function symbol $f : c \rightarrow c'$, if $p \approx_C q : c' \rightarrow c''$, then $f.p \approx_C f.q$.

We call $\approx_C$ the **C -provable equality relation** on paths, and we say that p_0 and p_1 are C -provably equal if $p_0 \approx_C p_1$.

Remark A.1. Frequently we will use structural induction on the rules 1-6 in Definition 2.3 to prove statements of the form “for all paths p and q , if $p \approx_C q$, then $P(p, q)$ ” where $P(p, q)$ is some property of p and q . To write such a proof, we simply need to show that for every inference rule, if the pairs of paths in the antecedent satisfy P , then the pair of paths in the conclusion satisfies P as well. For example, for rule 6, we would show that for all parallel C -paths p and q and C -function symbols f composable with p and q , if $p \approx_C q$ and $P(p, q)$, then $P(f.p, f.q)$.

Notice that if the property $P(p, q)$ is itself an equivalence relation, then the verifications corresponding to rules 2-4 are satisfied automatically.

Lemma 2.7. Given a category presentation C , $\approx_C$ is the smallest congruence containing C_E . In other words, the relations $\sim_{C_E}$ and $\approx_C$ coincide.

Proof. By rule 1, $\approx_C$ contains C_E , while by rules 2, 3, and 4, $\approx_C$ is an equivalence relation. To establish that $\approx_C$ is a congruence it remains to prove that $p \approx_C q$ implies p and q have the same domain and codomain (which follows straightforwardly by induction) and that $\approx_C$ is preserved by composition. To this end, we extend rules 5 and 6 to the case where f is a path, this time by induction on f . Then whenever $f_1 \approx_C f_2$ and $g_1 \approx_C g_2$ and the composition $f_1.g_1$ exists, we have $f_1.g_1 \approx_C f_1.g_2$ and $f_1.g_2 \approx_C f_2.g_2$; then we conclude $f_1.g_1 \approx_C f_2.g_2$ by rule 4. Now let $\approx$ be a congruence containing C_E . A straightforward application of induction proves that $p \approx_C q$ implies $p \approx q$. So $\approx_C$ is the smallest congruence containing C_E . $\square$

Lemma 2.10. Given a morphism $F : C \rightarrow C'$ of category presentations, if $p_0 \approx_C p_1$, then $F(p_0) \approx_{C'} F(p_1)$.

Proof. Straightforward induction, see Rem. A.1. $\square$

⁸That is to say, there are no pairs (p, q) in $\approx_C$ other than what can be derived in finitely many steps with the rules 1-6.

B Proofs for Section 3 (Profunctor Presentations)

For the subsequent proofs of this section and the next, we must prove many statements of the form $s \approx_P t \Rightarrow A(s, t)$ for terms s and t in a (C, D) -uncurried profunctor presentation P , and where $A(s, t)$ is a statement involving s and t . We could attempt to do this by proving $s \approx_{|P|} t \Rightarrow A(s, t)$ by induction, but this would require us to extend $A(s, t)$ to arbitrary paths s and t which are not necessarily cross-paths. The following result provides a useful inductive characterization for $\approx_P$ in its own right.

Lemma B.1. Given a (C, D) -uncurried profunctor presentation P , the relation $\approx_P$ coincides with the relation $\sim$ on cross-paths of P defined inductively by the following inference rules:

1. For any equation $s =_P t$, $s \sim t$.
2. For any cross-path t of P , $t \sim t$.
3. If $s \sim t$, then $t \sim s$.
4. If $r \sim s$ and $s \sim t$, then $r \sim t$.
5. For any $(g : d \rightarrow d') \in \text{Fun}(D)$ and cross-paths $s, t : c \rightarrow d$, if $s \sim t$, then $s.g \sim t.g$.
6. For any $(f : c' \rightarrow c) \in \text{Fun}(C)$ and cross-paths $s, t : c \rightarrow d$, if $s \sim t$, then $f.s \sim f.t$.
7. If $t : c \rightarrow d$ is a cross-path of P and $g =_D g' : d \rightarrow d'$, then $t.g \sim t.g'$.
8. If $t : c \rightarrow d$ is a cross-path of P and $f =_C f' : c' \rightarrow c$, then $f.t \sim f'.t$.

Proof. A straightforward induction on the structure of $\sim$ (analogous to Remark A.1) shows that $\sim$ is contained in $\approx_{|P|}$, and hence in $\approx_P$. To show the converse, let us show that if $s \approx_{|P|} t$, then either s and t are C -paths and $s \approx_C t$, s and t are D -paths and $s \approx_D t$, or s and t are cross-paths of I and $s \sim t$. Let $A(s, t) = \text{“either } s \text{ and } t \text{ are } C\text{-paths and } s \approx_C t, s \text{ and } t \text{ are } D\text{-paths and } s \approx_D t, \text{ or } s \text{ and } t \text{ are cross-paths of } I \text{ and } s \sim t\text{”}$. Now we will prove that $s \approx_{|P|} t$ implies $A(s, t)$ by induction on $\approx_{|P|}$, given by Definition 2.3. If $s =_{|P|} t$, then either $s =_C t$ if they are C -paths, $s =_D t$ if they are D -paths, or $s =_P t$ if they are cross-paths, thus $A(s, t)$ holds. Since A is an equivalence relation, rules 2-4 preserve A (see Remark A.1).

For rule 5 of Definition 2.3, suppose that $f : p' \rightarrow p''$ and $s, t : p \rightarrow p'$ are $|P|$ -paths such that $s \approx_{|P|} t$ and $A(s, t)$. If p'' is a sort of C , then clearly $s.f \approx_C t.f$ and $A(s.f, t.f)$ holds. If p is a sort of D , then clearly $s.f \approx_D t.f$ and $A(s.f, t.f)$ holds. If p is a sort of C and p' and p'' are sorts of D , then s and t are cross-paths and $s \sim t$. Thus it follows that $s.f \sim t.f$ by rule 5 above, hence $A(s.f, t.f)$. Finally, we consider the case where p and p' are sorts of C and p'' is a sort of D . In this case we have $s \approx_C t$ and f is a cross-path. We show that $s.f \sim t.f$, and hence $A(s.f, t.f)$, by a nested induction on $s \approx_C t$. More specifically, we show that whenever $s \approx_C t : c' \rightarrow c$, $B(s, t)$ holds, where $B(s, t) := \text{“for all cross-paths } f : c \rightarrow d \text{ we have } s.f \sim t.f\text{”}$. If $s =_C t$, then $s.f \sim t.f$ follows for all cross-paths f by rule 8 above. Clearly B is an equivalence relation, so the verifications for rules 2-4 of Def. 2.3 are automatic. For rule 5 of Def. 2.3, we assume that $B(s, t)$ holds for $s, t : c'' \rightarrow c'$ with $s \approx_C t$ and we let $(u : c' \rightarrow c) \in \text{Fun}(C)$. Then for any cross-path $f : c \rightarrow d$, we obtain $s.u.f \sim t.u.f$ by instantiating B on the cross-path $u.f$. For rule 6 of Def. 2.3, we assume that $B(s, t)$ holds for $s, t : c' \rightarrow c$ with $s \approx_C t$ and we let $(u : c'' \rightarrow c') \in \text{Fun}(C)$. Then for any cross-path $f : c \rightarrow d$ we obtain $u.s.f \sim u.t.f$ by applying rule 6 above to the instantiation of $B(s, t)$ at f .

The verification of rule 6 of Def. 2.3 is dual to the verification of rule 5 that we have just completed. $\square$

The following lemma follows immediately as an important special case of Lemma B.1.

Lemma B.2. Given a C -instance presentation I , the relation $\approx_I$ coincides with the relation $\sim$ on terms of I defined inductively by the following inference rules:

1. For any equation $s =_I t$, $s \sim t$.
2. For any term t of I , $t \sim t$.
3. If $s \sim t$, then $t \sim s$.
4. If $r \sim s$ and $s \sim t$, then $r \sim t$.
5. For any C -function symbol $f : c \rightarrow d$ and terms $(s, t : c)$, if $s \sim t$, then $s.f \sim t.f : d$.
6. If $t : c$ is a term of I and $(p =_C q : c \rightarrow d)$, then $t.p \sim t.q : d$.

Lemma 3.22. Given curried profunctor presentations $P : C \nrightarrow D$ and $Q : D \nrightarrow E$, the operation $\otimes$ respects provable equality. In other words, if $(s \approx_{P(c)} s' : d)$ and $(t \approx_{Q(d)} t' : e)$, then $(s \otimes t \approx_{(P \otimes Q)(c)} s' \otimes t' : e)$.

Proof. First let us prove the following statements:

1. if $(t \approx_{Q(d)} t' : e)$, then for all $(p : d) \in \text{Gen}(P(c))$, $(p \otimes t \approx_{(P \otimes Q)(c)} p \otimes t' : e)$.
2. if $(t \approx_{Q(d)} t' : e)$, then for all $(s : d) \in \text{Term}(P(c))$, $(s \otimes t \approx_{(P \otimes Q)(c)} s \otimes t' : e)$.
3. if $(s \approx_{P(c)} s' : d)$, then for all $(t : e) \in \text{Term}(Q(d))$, $(s \otimes t \approx_{(P \otimes Q)(c)} s' \otimes t : e)$.

Given the definition of the equations in $P \otimes Q$, it is straightforward to prove (1) by induction on $\approx_{Q(d)}$ using Lemma B.2. Let us now consider (2). If $(s : d) \in \text{Term}(P(c))$, then $s \equiv p.g$ for some generator p and D -path $g : d' \rightarrow d$. Thus $s \otimes t \equiv (p \otimes Q(g)(t))$ and $s \otimes t' \equiv (p \otimes Q(g)(t'))$. Since Q is a curried profunctor presentation and $t \approx_{Q(d)} t'$, we have $Q(g)(t) \approx_{Q(d')} Q(g)(t')$ by Lemma 2.10, and thus by (1), it follows that $s \otimes t \equiv (p \otimes Q(g)(t)) \approx_{(P \otimes Q)(c)} p \otimes Q(g)(t') \equiv s \otimes t'$.

Now let us prove (3) by induction on $\approx_{P(c)}$ using Lemma B.2. If $(s =_{P(c)} s' : d)$ and $(t \equiv q.h : e) \in \text{Term}(Q(d))$, then there is an equation $(s \otimes q =_{(P \otimes Q)(c)} s' \otimes q : e)$, hence $s \otimes t \equiv (s \otimes q).h \approx_{(P \otimes Q)(c)} (s' \otimes q).h \equiv s' \otimes t$ by rule 6 of Lemma B.2. Rules 2-4 clearly follow from the fact that the property in the consequent of (3) is reflexive, symmetric and transitive.

For Rule 5, we want to show that if $s, s' \in \text{Term}(P(c))$, $(s \approx_{P(c)} s' : d)$ and for all $(t : e) \in \text{Term}(Q(d))$, $(s \otimes t \approx_{(P \otimes Q)(c)} s' \otimes t : e)$, then for any D -path $g : d \rightarrow d'$ and term $(t : e) \in \text{Term}(Q(d'))$ we have $(s.g \otimes t \approx_{(P \otimes Q)(c)} s'.g \otimes t : e)$. But this is equivalent to showing that $(s \otimes Q(g)(t) \approx_{(P \otimes Q)(c)} s' \otimes Q(g)(t))$, which follows by the inductive hypothesis.

Finally, for Rule 6, suppose that $(g =_D g' : d \rightarrow d')$. Then $Q(g) \approx Q(g')$ since Q is a curried profunctor presentation. Thus for every $(s : d) \in \text{Term}(P(c))$ and $(t : e) \in \text{Term}(Q(d'))$, $s \otimes Q(g)(t) \approx_{(P \otimes Q)(c)} s \otimes Q(g')(t)$ by (2) above, which is to say that $s.g \otimes t \approx_{(P \otimes Q)(c)} s.g' \otimes t$.

Having proven statements (1), (2) and (3), the result follows by transitivity of $\approx_{(P \otimes Q)(c)}$. $\square$

Lemma 3.24. Given a C -function symbol $f : c \rightarrow c'$ in C , the function $(P \otimes Q)(f)$ is a morphism of E -instances. Furthermore $P \otimes Q$ is a well-defined curried profunctor presentation from C to E .

Proof. First notice that $(P \otimes Q)(f)(s \otimes t) \equiv P(f)(s) \otimes t$ for all $(f : c \rightarrow c') \in \text{Path}(C)$, $(s : d) \in \text{Term}(P(c))$, and $(t : e) \in \text{Term}(Q(d))$, as can be checked by a simple computation.

Now given a C -function symbol $f : c \rightarrow c'$, to show that $(P \otimes Q)(f)$ is an E -instance morphism, we must show that it sends equations in $(P \otimes Q)(c')$ to provable equalities in $(P \otimes Q)(c)$. Since equations in $(P \otimes Q)(c')$ have two possible forms, we consider them one by one. If $(s =_{P(c)} s' : d)$ and $(q : e) \in$

$\text{Gen}(P(d))$, then $(P(f)(s) \otimes q \approx_{(P \otimes Q)(c)} P(f)(s') \otimes q)$ by Lemma 3.22. On the other hand, if $(p : d) \in \text{Gen}(P(c))$ and $(t =_{Q(d)} t' : d)$, then $(P(f)(p) \otimes t \approx_{(P \otimes Q)(c)} P(f)(p) \otimes t')$ again by Lemma 3.22.

Secondly, we must show that whenever $(f =_C f' : c \rightarrow c')$ is an equation between C -paths, we have $(P \otimes Q)(f) \approx (P \otimes Q)(f')$. Thus this reduces to showing that $P(f)(p) \otimes q \approx_{(P \otimes Q)(c)} P(f')(p) \otimes q$ for all generators $(p \otimes q) \in \text{Gen}((P \otimes Q)(c'))$. But this follows directly from the fact that $P(f) \approx P(f')$ and Lemma 3.22. $\square$

Lemma 3.25. The assignment $(P, Q) \mapsto P \otimes Q$ defines a functor

$$\otimes : \mathbf{Curr}(C, D) \times \mathbf{Curr}(D, E) \rightarrow \mathbf{Curr}(C, E).$$

Proof. Suppose that $P, P' : C \rightarrow D$ and $Q, Q' : D \rightarrow E$ are curried profunctor presentations and $\phi : P \rightarrow P'$ and $\psi : Q \rightarrow Q'$ are morphisms of curried profunctor presentations. Given $c \in C$, let $(\phi \otimes \psi)_c : (P \otimes Q)(c) \rightarrow (P' \otimes Q')(c)$ denote the map of E -instance presentations defined by

$$(\phi \otimes \psi)_c(p \otimes q) := (\phi_c(p) \otimes \psi_d(q))$$

for all $(p : d) \in \text{Gen}(P(c))$ and $(q : e) \in \text{Gen}(Q(d))$. In what follows we make repeated use of Lemma 3.22 without stating it explicitly.

First, notice that that

$$(\phi \otimes \psi)_c(s \otimes t) \approx_{(P' \otimes Q')(c)} \phi_c(s) \otimes \psi_d(t). \quad (4)$$

for all $(s : d) \in \text{Term}(P(c))$ and $(t : e) \in \text{Term}(Q(d))$. Indeed, expand $s \equiv p.g$ where $(p : d') \in \text{Gen}(P(c))$ and $g : d' \rightarrow d$ is a D -path, and $Q(g)(t) \equiv q.h$. Then

$$(\phi \otimes \psi)_c(s \otimes t) \equiv (\phi \otimes \psi)_c(p \otimes q.h) \equiv \phi_c(p) \otimes \psi_{d'}(q).h \approx_{(P' \otimes Q')(c)} \phi_c(p) \otimes Q'(g)(\psi_d(t)) \equiv \phi_c(s) \otimes \psi_d(t)$$

where we have used “naturality” of ψ as expressed by diagram (3). The fact that $(\phi \otimes \psi)_c$ is an E -instance presentation morphism then follows directly from (4) by looking at the specific form of any equation in $P \otimes Q$ and remembering that both ϕ_c and ψ_d are morphisms of instance presentations.

Finally, let us show that the naturality condition expressed by diagram (3) holds for the family of morphisms $\{(\phi \otimes \psi)_c\}_{c \in C}$. Given a function symbol $f : c \rightarrow c'$ in C and a generator $(p \otimes q : e) \in \text{Gen}((P \otimes Q)(c'))$, where $(p : d) \in \text{Gen}(P(c'))$,

$$\begin{aligned} (\phi \otimes \psi)_c((P \otimes Q)(f)(p \otimes q)) &\equiv \phi_c(P(f)(p)) \otimes \psi_d(q) && \text{(def. of } \phi \otimes \psi) \\ &\approx_{(P' \otimes Q')(c)} P'(f)(\phi_{c'}(p)) \otimes \psi_d(q) && \text{(naturality of } \phi) \\ &\equiv (P' \otimes Q')(f)(\phi_{c'}(p) \otimes \psi_d(q)) && \text{(def. of } (P' \otimes Q')(f)) \\ &\equiv (P' \otimes Q')(f)((\phi \otimes \psi)_{c'}(p \otimes q)). && \text{(def. of } \phi \otimes \psi) \end{aligned}$$

$\square$

Theorem 3.26. Given curried profunctor presentations $P : C \rightarrow D$ and $Q : D \rightarrow E$, there is an isomorphism

$$\mu^{P, Q} : \langle P \rangle \odot \langle Q \rangle \rightarrow \langle P \otimes Q \rangle$$

of profunctors, natural in P and Q . Furthermore, if $P : C \rightarrow D$ and $Q : D \rightarrow E$ are finite curried profunctor presentations, then $P \otimes Q$ is a finite curried profunctor presentation. In particular, composition of profunctors preserves finite curried presentability.

Proof. It is obvious by construction that if $P : C \rightarrow D$ and $Q : D \rightarrow E$ are finite curried profunctor presentations, then $(P \otimes Q)$ is a finite curried profunctor presentation. Thus all we need to do is show that there is an isomorphism of profunctors $\mu^{P,Q} : (P) \odot (Q) \cong (P \otimes Q)$ natural in P and Q . We consider both profunctors as functors $(C)^{\text{op}} \times (D) \rightarrow \mathbf{Set}$. Hence, to give this isomorphism is to give a natural isomorphism, whose components are bijections $\mu_{(c,e)}^{P,Q} : ((P) \odot (Q))(c, e) \rightarrow (P \otimes Q)(c, e)$ for each $c \in C$ and $e \in E$. Moreover, we work with the concrete description of the sets $((P) \odot (Q))(c, e)$ given by (2), hence these functions are defined on equivalence classes $\langle [s], [t] \rangle$ where $[s] \in (P)(c, d)$ and $[t] \in (Q)(d, e)$ for some $d \in D$. Notice that in this formulation of profunctors as functors, their action on morphisms are given by pre- and postcomposition, i.e. given $f : c' \rightarrow c$ a C -path and $k : e \rightarrow e'$ an E -path,

$$\begin{aligned} ((P) \odot (Q))([f], [k])(\langle [s], [t] \rangle) &= \langle (P)([f])([s]), [t.k] \rangle \\ &= \langle [P(f)(s)], [t.k] \rangle \quad \text{and} \\ (P \otimes Q)([f], [k])([(p \otimes q).h]) &= [(P \otimes Q)(f)(p \otimes q).h.k] \\ &= [((P)(f)(p) \otimes q).h.k]. \end{aligned}$$

Let us then define, for each $c \in C$ and $e \in E$, the map $\mu_{(c,e)}^{P,Q}$ by the formula

$$\mu_{(c,e)}^{P,Q}(\langle [s], [t] \rangle) := [s \otimes t]$$

for all $d \in D$, $(s : d) \in \text{Term}(P(c))$ and $(t : e) \in \text{Term}(Q(d))$. We will now prove that this is a well defined function. Notice that the domain of $\mu_{(c,e)}^{P,Q}$ is defined by two nested quotients by equivalence relations. The situation is clarified by the following diagram

$$\begin{array}{ccc} S_{(c,e)} & \xrightarrow{F} & (P \otimes Q)(c, e) \\ q_1 \downarrow & \nearrow \tilde{F} & \uparrow \mu_{(c,e)}^{P,Q} \\ \tilde{S}_{(c,e)} & \xrightarrow{q_2} & ((P) \odot (Q))(c, e) \end{array} \quad \begin{array}{ccc} (s, t) & \xrightarrow{F} & [s \otimes t] \\ q_1 \downarrow & & \uparrow \\ ([s], [t]) & \xrightarrow{q_2} & \langle [s], [t] \rangle \end{array}$$

where $S_{(c,e)} := \sum_{(s:d) \in \text{Term}(P(c))} \{(t : e) \in \text{Term}(Q(d))\}$ and $\tilde{S}_{(c,e)} := \sum_{d \in D} (P)(c, d) \times (Q)(d, e)$, and the functions F , q_1 and q_2 are given by the mappings on the right. We will first prove that F is constant on the fibers of q_1 , hence inducing a function $\tilde{F}$ making the upper triangle in the diagram commute, and then that $\tilde{F}$ is constant on the fibers of q_2 , hence inducing our desired function $\mu_{(c,e)}^{P,Q}$. Both are relatively straightforward.

The first step amounts to showing that the expression $[s \otimes t]$ is a well defined function of the pair $([s], [t])$. Let $(s : d), (s' : d') \in \text{Term}(P(c))$, $(t : e) \in \text{Term}(Q(d))$ and $(t' : e) \in \text{Term}(Q(d'))$ be such that $([s], [t]) = ([s'], [t'])$. Then it must be the case that $d' = d$, $s \approx_{P(c)} s'$ and $t' \approx_{Q(d)} t$, from which $s \otimes t \approx_{(P \otimes Q)(c)} s' \otimes t'$ by Lemma 3.22.

For the second step, let $R \subseteq \tilde{S}_{(c,e)} \times \tilde{S}_{(c,e)}$ be the binary relation given by $([s], [t]) R ([s'], [t'])$ if and only if $\tilde{F}([s], [t]) = [s \otimes t] = [s' \otimes t'] = \tilde{F}([s'], [t'])$. Since the relation “ $q_2([s], [t]) = q_2([s'], [t'])$ ” is precisely the relation $\simeq$ from (2), and since $\simeq$ is defined as the smallest equivalence relation containing $\simeq_0$, it is enough to show that R is an equivalence relation containing $\simeq_0$. Since R is clearly an equivalence relation, it only remains to show that R contains $\simeq_0$. Given $[s] : c \rightarrow d$, $[s'] : c \rightarrow d'$, $[t] : d \rightarrow e$ and $[t'] : d' \rightarrow e$, suppose $([s], [t]) \simeq_0 ([s'], [t'])$. Then there exists a D -path $f : d \rightarrow d'$ such that $[s.f] = [s']$ and $[Q(f)(t')] = [t]$. Since $s.f \otimes t' \equiv s \otimes Q(f)(t')$, we have $\tilde{F}([s], [t]) = \tilde{F}([s], [Q(f)(t')]) = [s \otimes Q(f)(t')] = [s.f \otimes t'] = \tilde{F}([s.f], [t']) = \tilde{F}([s'], [t'])$.

Thus we conclude that $\mu_{(c,e)}^{P,Q}$ is well defined. Meanwhile, naturality of $\mu^{P,Q}$ amounts to the fact that, for all C -paths $f : c' \rightarrow c$ and E -paths $k : e \rightarrow e'$, we have the following:

$$\begin{array}{ccc} \langle [s], [t] \rangle & \xrightarrow{\quad\quad\quad} & \langle [P(f)(s)], [t.k] \rangle \\ \mu_{(c,e)}^{P,Q} \downarrow & & \downarrow \mu_{(c',e')}^{P,Q} \\ [s \otimes t] & \xrightarrow{\quad\quad\quad} & [(P \otimes Q)(f)(s \otimes t).k] = [P(f)(s) \otimes t.k]. \end{array}$$

We have proved that $\mu^{P,Q}$ is a natural transformation. Let us now prove that it is an isomorphism. Fixing again c and e , and setting $\alpha := \mu_{(c,e)}^{P,Q}$ for the remainder of this proof, we define a map backwards $\beta : (P \otimes Q)(c, e) \rightarrow ((P) \odot (Q))(c, e)$ as follows. If $x \in (P \otimes Q)(c, e)$, then it is of the form $x = [r]$ for some term $r \equiv (p \otimes q).h$ where $(p : d) \in \text{Gen}(P(c))$ and $(q.h : e) \in \text{Term}(Q(d))$. We then define $\beta(x) := \langle [p], [q.h] \rangle$. Let us show that β is well defined by induction on $\approx_{(P \otimes Q)(c)}$.

For the base case, consider an equation $r =_{(P \otimes Q)(c)} r'$. There are two possibilities:

- $r \equiv s \otimes q$, $r' \equiv s' \otimes q$ where $s =_{P(c)} s' : d$ and $(q : e) \in \text{Gen}(Q(d))$. In this case, let $s \equiv p.g$ and $s' \equiv p'.g'$, hence $r \equiv p \otimes Q(g)(q)$ and $r' \equiv p' \otimes Q(g')(q)$. Then

$$\beta([r]) = \langle [p], [Q(g)(q)] \rangle = \langle [p.g], [q] \rangle = \langle [p'.g'], [q] \rangle = \langle [p'], [Q(g')(q)] \rangle = \beta([r']).$$

- $r \equiv p \otimes t$, $r' \equiv p \otimes t'$ where $(p : d) \in \text{Gen}(P(c))$ and $t =_{Q(d)} t' : e$. In this case we conclude immediately that $\beta([r]) = \langle [p], [t] \rangle = \langle [p], [t'] \rangle = \beta([r'])$.

Thus the base case is proved. Meanwhile, rules 2-4 of Lemma B.2 are immediate from the fact that the relation $\{((p \otimes q).h, (p' \otimes q').h') \mid \langle [p], [q.h] \rangle = \langle [p'], [q'.h'] \rangle\}$ is an equivalence relation. For rule 5, suppose that $(r \approx_{(P \otimes Q)(c)} r' : e)$, $k : e \rightarrow e'$ is an E -path and $\beta([r]) = \beta([r'])$. Let $r \equiv (p \otimes q).h$ and $r' \equiv (p' \otimes q').h'$ so that $\langle [p], [q.h] \rangle = \langle [p'], [q'.h'] \rangle$. Then

$$\langle [p], [q.h.k] \rangle = ((P) \odot (Q))(1, [k])(\langle [p], [q.h] \rangle) = ((P) \odot (Q))(1, [k])(\langle [p'], [q'.h'] \rangle) = \langle [p'], [q'.h'.k] \rangle.$$

Rule 6 follows by similar reasoning, from the fact that if $(k =_E k' : e \rightarrow e')$, then $[k] = [k']$ in (E) and hence $((P) \odot (Q))(1, [k]) = ((P) \odot (Q))(1, [k'])$. Thus β is well defined.

Now it is easy to show that α and β are inverse to each other. Indeed, $(\beta \circ \alpha)(\langle [p.g], [t] \rangle) = \beta([p.g \otimes t]) = \langle [p], [Q(g)(t)] \rangle = \langle [p.g], [t] \rangle$. Similarly, $(\alpha \circ \beta)([(p \otimes q).h]) = \alpha(\langle [p], [q.h] \rangle) = [p \otimes q.h] = [(p \otimes q).h]$.

Thus $\alpha = \mu_{(c,e)}^{P,Q}$ is a bijection for every (c, e) , and therefore $\mu^{P,Q}$ defines an isomorphism of profunctors.

Finally, we must show that $\mu : (-) \odot (-) \Rightarrow (- \otimes -)$ is a natural transformation between functors $\mathbf{Curr}(C, D) \times \mathbf{Curr}(D, E) \rightarrow \mathbf{Prof}(\langle C \rangle, \langle E \rangle)$. Let $\phi : P \rightarrow P'$ and $\psi : Q \rightarrow Q'$ be morphisms of curried profunctor presentations $P, P' : C \rightarrow D$ and $Q, Q' : D \rightarrow E$. Then naturality amounts to the verification

$$\begin{aligned} ((\phi \otimes \psi) \circ \mu^{P,Q})(\langle [s], [t] \rangle) &= (\phi \otimes \psi)([s \otimes t]) \\ &= [\phi_c(s) \otimes \psi_d(t)] \\ &= \mu^{P',Q'}(\langle [\phi_c(s)], [\psi_d(t)] \rangle) \\ &= (\mu^{P',Q'} \circ ((\phi) \odot (\psi)))(\langle [s], [t] \rangle). \end{aligned}$$

□

C Proofs for Section 4 (Curryable Profunctor Presentations)

Proposition 4.2. Given category presentations C and D , the construction given in Definition 4.1 defines a functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ which preserves provable equality of morphisms and makes the following diagram commute up to isomorphism

$$\begin{array}{ccc}
 \mathbf{Curr}(C, D) & \xrightarrow{\overline{(-)}} & \mathbf{UnCurr}(C, D) \\
 & \searrow \scriptstyle{(-)} \quad \swarrow \scriptstyle{(-)} & \\
 & \mathbf{Prof}(\langle C \rangle, \langle D \rangle) &
 \end{array}$$

Furthermore, $\overline{(-)}$ restricts to a functor $\overline{(-)} : \mathbf{FinCurr}(C, D) \rightarrow \mathbf{FinUnCurr}(C, D)$.

Proof. We give a sketch of proof. First note that $\overline{(-)} : \text{Term}(P(c)) \rightarrow \text{CPath}(\overline{P})$ preserves provable equality, as can be shown by an easy induction. This is the essential ingredient needed to show that whenever $F : P \rightarrow Q$ is a morphism of curried profunctor presentations, $\overline{F}$ is a morphism of uncurried profunctor presentations; and that the functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ preserves provable equality of morphisms.

The commutation up to isomorphism of the triangle also uses this same ingredient, as well as the fact that whenever P is a curried profunctor presentation, $\overline{P}$ is nongenerative and conservative (see Def. 4.5), which will be shown in Thm. 4.11. $\square$

Lemma C.1. Let $P : C \rightarrowtail D$ be a curried profunctor presentation, and let $f : c' \rightarrow c$ be a C -path and $t \in \text{Term}(P(c))$. Then

$$f.\bar{t} \approx_{\overline{P}} \overline{P(f)(t)}.$$

Proof. We prove this by induction on f . If $f \equiv 1_c$, then the conclusion follows trivially. If the conclusion holds for a given $f : c' \rightarrow c$ and $a : c'' \rightarrow c'$ is a function symbol, then $a.f.\bar{t} \approx_{\overline{P}} a.\overline{P(f)(t)}$. But $P(f)(t) \equiv q.h$ for some generator q and D -path h . Thus $a.q.h \approx_{\overline{P}} \overline{P(a)(q.h)}$ by the equations of $\overline{P}$, and since $\overline{P(a)(q.h)} \equiv \overline{P(a.f)(t)}$, the result follows. $\square$

Lemma 4.7. Let $P : C \rightarrowtail D$ be a nongenerative uncurried profunctor presentation. Then for every cross-path t of P there exists a right cross-path r such that $t \approx_P r$.

Proof. Write $t \equiv f.p.g$ where $(p : c \rightarrow d) \in \text{Fun}(P)$. Clearly the lemma holds when $f = 1_c$. Now suppose the lemma holds for a C -path $f : c' \rightarrow c$, and let $a : c'' \rightarrow c'$ be a function symbol. By the inductive hypothesis, there exists a right cross-path $r \equiv p'.g'$, where p' is a generator, such that $f.p.g \approx_P p'.g'$. Thus $a.f.p.g \approx_P a.p'.g'$. Since P is nongenerative and $a.p'$ is a short left cross-path, there exists a right cross-path r' such that $a.p' \approx_P r'$, so $a.f.p.g \approx_P r'.g'$. $\square$

Lemma 4.9. Given an uncurried profunctor presentation $P : C \rightarrowtail D$, for every $c \in C$, there is an obvious inclusion functor⁹ $\iota^c : \langle P^c \rangle \rightarrow \langle P \rangle$. Thus P is nongenerative if and only if for every $c \in C$, ι^c is full on cross-morphisms. Similarly P is conservative if and only if for every $c \in C$, ι^c is faithful.

⁹Where here we think of $\langle P^c \rangle$ as a profunctor $\langle P^c \rangle : * \rightarrowtail D$.

Proof. Suppose P is nongenerative. Given an element $[t] \in \langle P \rangle(c, d)$, by Lemma 4.7 there exists a right cross-path r in P such that $t \approx_P r$. Since r is a right cross-path starting at c , it is a term in P^c , and thus $[t] = \iota^c([r])$. Therefore ι^c is full on cross-morphisms. Conversely, suppose ι^c is full on cross-morphisms for all c . Then given a short left cross-path t of P , there exists a term r in P^c such that $[t] = \iota^c([r])$. Since r is a term in P^c , it is a right cross-path in P . This implies that $t \approx_P r$, and since t was arbitrary, P is nongenerative.

Now suppose P is conservative. Given $[r], [r'] \in \langle P^c \rangle(d)$ with $\iota^c([r]) = \iota^c([r'])$, by conservativity we have $r \approx_{P^c} r'$, so $[r] = [r']$. Conversely, suppose ι^c is faithful. If r and r' are right cross-paths in P with $r \approx_P r'$, then we have $\iota^c([r]) = \iota^c([r'])$. Since ι^c is faithful, this implies that $r \approx_{P^c} r'$. $\square$

Theorem 4.11. Given category presentations C and D , the functor $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{UnCurr}(C, D)$ takes values in curryable profunctor presentations and rightward morphisms. Furthermore the corestrictions $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Crble}(C, D)$ and $\overline{(-)} : \mathbf{FinCurr}(C, D) \rightarrow \mathbf{FinCrble}(C, D)$ are equivalences of categories which preserve and reflect provable equality.

Proof. Suppose that $P : C \rightarrowtail D$ is a curried profunctor presentation. Then $\overline{P}$ is nongenerative by construction. We would like to show that $\overline{P}$ is also conservative by induction using Lemma B.1; however, to do this, we must show a statement implying conservativity which involves an arbitrary pair of provably equal $\overline{P}$ -cross-paths, not just right cross-paths, as in the definition of conservativity. Accordingly, we will show that whenever $f.\bar{t} \approx_{\overline{P}} f'.\bar{t}'$ is a provable equality of cross-paths, written such that $\bar{t}$ and $\bar{t}'$ are right cross-paths, we have $P(f)(t) \approx_{P(c)} P(f')(t')$. Once we show this, it follows *a fortiori* that whenever $\bar{t} \approx_{\overline{P}} \bar{t}'$ is a provable equality of right cross-paths, $t \approx_{P(c)} t'$, or, what is the same thing, $\bar{t} \approx_{\overline{P}^c} \bar{t}'$.

If $f.\bar{t} =_{\overline{P}} f'.\bar{t}'$, then either f and f' are both trivial, in which case the conclusion follows immediately, or f is a function symbol and f' is trivial, in which case $t' \equiv P(f)(t)$, in which case the conclusion follows by reflexivity of $\approx_{P(c)}$. Rules 2-4 of Lemma B.1 are verified automatically since our conclusion is an equivalence relation. The verifications for rules 5 and 7 follow immediately from the fact that $\approx_{P(c)}$ satisfies rules 5 and 6 of Lemma B.2.

For rule 6 of Lemma B.1, suppose that $f.\bar{t} \approx_{\overline{P}} f'.\bar{t}' : c \rightarrow d$ and $P(f)(t) \approx_{P(c)} P(f')(t')$, and let $(g : c' \rightarrow c) \in \text{Fun}(C)$. Then $P(g.f)(t) \equiv P(g)(P(f)(t)) \approx_{P(c')} P(g)(P(f')(t')) \equiv P(g.f')(t')$ because $P(g)$ is a D -instance presentation morphism.

For rule 8 of Lemma B.1, suppose that $f.\bar{t} \approx_{\overline{P}} f'.\bar{t}' : c \rightarrow d$ and $P(f)(t) \approx_{P(c)} P(f')(t')$, and let $g =_C g' : c' \rightarrow c$. Then $P(g.f)(t) \equiv P(g)(P(f)(t)) \approx_{P(c')} P(g)(P(f')(t')) \equiv P(g.f')(t')$ because $P(g) \approx P(g')$, since P is a curried presentation.

Hence $\overline{P}$ is conservative, and therefore curryable. If $F : P \rightarrow P'$ is a morphism of curried profunctor presentations, then clearly $\overline{F}$ sends profunctor function symbols to right cross-paths in $\overline{P}'$, so $\overline{F}$ is rightward, and $\overline{(-)}$ corestricts as required.

Now let us show that $\overline{(-)} : \mathbf{Curr}(C, D) \rightarrow \mathbf{Crble}(C, D)$ is an equivalence. Suppose that $Q : C \rightarrowtail D$ is a curryable profunctor presentation. We must construct a curried profunctor presentation P and an isomorphism $\overline{P} \cong Q$. First let us define $P : C \rightarrowtail D$ sortwise by $P(c) := Q^c$. Since Q is nongenerative, for each $(f : c \rightarrow c') \in \text{Fun}(C)$ and $p \in \text{Gen}(P(c'))$ there exists a right cross-path r such that $r \approx_Q f.p$. Using the Axiom of Choice, choose such an r , and set $P(f)(p) := r$. Thus we have $P(f)(p) \approx_Q f.p$, and thus holds for paths f as well by induction on f . Let us show that $P(f) : P(c') \rightarrow P(c)$ is a D -instance morphism. Given an equation $(p.g =_{P(c')} p'.g')$, we have

$$P(f)(p.g) \approx_Q f.p.g \approx_Q f.p'.g' \approx_Q P(f)(p'.g')$$

so by conservativity, $P(f)(p.g) \approx_{Q^c} P(f)(p'.g')$ which is equivalent to $P(f)(p.g) \approx_{P(c)} P(f)(p'.g')$. Finally, given an equation $(f =_c f' : c \rightarrow c')$ and a generator $p \in \text{Gen}(P(c'))$, we have $P(f)(p) \approx_Q f.p \approx_Q f'.p \approx_Q P(f')(p)$, so by conservativity, $P(f)(p) \approx_{Q^c} P(f')(p)$, or equivalently $P(f)(p) \approx_{P(c)} P(f')(p)$, which implies that P is a well defined curried profunctor presentation. Finally, we construct an isomorphism $\alpha : \bar{P} \cong Q$ by $\alpha(\bar{q}) := q$, with inverse β given by $\beta(q) := \bar{q}$. By construction α is a rightward morphism. Let us show that β sends equations to provable equalities. Given an equation $f.t =_Q f'.t' : c \rightarrow d$ where t and t' are right cross-paths, we have $P(f)(t) \approx_Q P(f')(t')$, so by conservativity $P(f)(t) \approx_{Q^c} P(f')(t')$, which is equivalent to $P(f)(t) \approx_{P(c)} P(f')(t')$, so $\beta(f.t) \equiv f.\bar{t} \approx_{\bar{P}} \bar{P}(f)(\bar{t}) \approx_{\bar{P}} \bar{P}(f')(t') \approx_{\bar{P}} f'.\bar{t}' \equiv \beta(f'.t')$ by Lemma C.1. Thus $\bar{(-)}$ is essentially surjective.

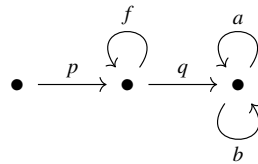
It is easy to see that $\bar{(-)}$ is faithful; let us show it is full. Let $P, P' : C \rightarrow D$ be curried profunctor presentations and let $G : \bar{P} \rightarrow \bar{P}'$ be a rightward morphism of curryable profunctor presentations. Thus for each $p \in \text{Gen}(P(c))$, $G(\bar{p})$ is a right cross-path $\bar{r}$. Thus we define $F : P \rightarrow P'$ by $F_c(p) := r$. Then for every generator we have $\bar{F}(\bar{p}) \equiv G(\bar{p})$, so $\bar{(-)}$ is full.

We have by Prop. 4.2 that $\bar{(-)}$ preserves provable equality; to see that it also reflects provable equality, suppose that $F, G : P \rightarrow Q$ are morphisms of curried presentations with $\bar{F} \approx \bar{G}$. Then for a generator $(p : c \rightarrow d)$ of P , $\bar{F}_c(p) \equiv \bar{F}(\bar{p}) \approx_{\bar{Q}} \bar{G}(\bar{p}) \equiv \bar{G}_c(p)$. Since $\bar{Q}$ is conservative, $\bar{F}_c(p) \approx_{\bar{Q}^c} \bar{G}_c(p)$, so $F_c(p) \approx_{Q(c)} G_c(p)$ and $P \approx Q$.

It follows easily that $\bar{(-)} : \mathbf{FinCurr} \rightarrow \mathbf{FinCrble}$ is also an equivalence, since $\bar{(-)}$ preserves finiteness and the curried presentation P constructed in our proof above of essential surjectivity is finite when Q is. $\square$

Proposition 4.13. There exist finite nongenerative uncurried profunctor presentations $P : C \rightarrow D$ and $Q : D \rightarrow E$ between finite category presentations such that $(P) \odot (Q)$ does not admit a finite uncurried presentation.

Proof. Consider the monoid presentations and bimodule presentations $M \xrightarrow{P} N \xrightarrow{Q} O$ defined by $M := \langle \emptyset \mid \emptyset \rangle$, $N := \langle f \mid \emptyset \rangle$, $O := \langle \emptyset \mid a, b \rangle$, $P(*) := \langle p \mid \emptyset \rangle_N$, and $Q(*) := \langle q \mid f.q = q.a, q = q.b \rangle_O$, as depicted in the following diagram:



Both P and Q are nongenerative, as can be easily checked.

One can show by induction (using Lemma B.1) that every instance of $\approx_P$ is of the form $p.f^n \approx_P p.f^n$ for some $n \in \mathbb{N}$. Thus the equivalence classes $p_n := [p.f^n]$ are distinct for each $n \in \mathbb{N}$ and $(P)(*, *) = \{p_n \mid n \in \mathbb{N}\}$.

Similarly, induction shows that every instance of $\approx_Q$ is of the form $f^n.q.w \approx_Q f^{n'}.q.w'$ satisfying $n + \#_a(w) = n' + \#_a(w')$, where $\#_a(w)$ is the number of times that a appears in the path w . Thus the equivalence classes $q_n := [q.a^n]$ are distinct for $n \in \mathbb{N}$ and $(Q)(*, *) = \{q_n \mid n \in \mathbb{N}\}$, with $(Q)(f, 1)(q_n) = q_{n+1}$, $(Q)(1, a)(q_n) = q_{n+1}$, and $(Q)(1, b)(q_n) = q_n$.

Now consider the composite profunctor $(P) \odot (Q) : (C) \rightarrow (E)$. The relation $\simeq_0$ as in (2) is then defined by $(p_m, q_{n+o}) \simeq_0 (p_{m+o}, q_n)$ for $m, n, o \geq 0$. Thus $(p_m, q_n) \simeq (p_{m'}, q_{n'})$ whenever $m + n = m' + n'$, and moreover, $\{((p_m, q_n), (p_{m'}, q_{n'})) \mid m + n = m' + n'\}$ is an equivalence relation so it is exactly $\simeq$. Hence

the equivalence classes $pq_n := \langle p_0, q_n \rangle$ are distinct for $n \in \mathbb{N}$ and $((P) \odot (Q))(*, *) = \{pq_n \mid n \in \mathbb{N}\}$, with $((P) \odot (Q))(1, a)(pq_n) = pq_{n+1}$ and $((P) \odot (Q))(1, b)(pq_n) = pq_n$.

Now suppose that R is a finite uncurried presentation and $K : (R) \cong (P) \odot (Q)$ is an isomorphism. Define the **order** $o(t)$ of a cross-path t of R as the unique n such that $K([t]) = pq_n$. Note that provably equal cross-paths of R have identical order. Moreover, since K is an isomorphism, the function $o : \text{CPath}(R) \rightarrow \mathbb{N}$ is surjective.

Since R_E is finite, let $N := \max\{o(t) \mid t =_R t'\}$ (we let $N := 0$ if R_E is empty). It is not hard to show by induction that whenever $t \approx_R t'$, either $o(t) \leq N$ or there exist $s, s' \in \text{CPath}(R)$ and $g \in \text{Fun}(O)$ such that $t \equiv s.g$ and $t' \equiv s'.g$. Let $t \in \text{CPath}(R)$ be the shortest cross-path such that $o(t) = N + 1$. Then we must have $[t.b] = (R)(1, b)([t]) = [t]$, so $t.b \approx_R t$. Since $o(t) > N$, there must be $s \in \text{CPath}(R)$ such that $t =_R s.b$. But then s is a shorter cross-path than t with $o(s) = N + 1$, contradicting the definition of t . $\square$

D Proofs for Section 5 (The Double Category of Curried Profunctor Presentations)

We include definitions of double categories and double functors for ease of reference and to fix notation for the proofs that follow.

Notation D.1. In what follows, given morphisms $f : A \rightarrow B$ and $g : C \rightarrow B$, let $A \times_{f, B, g} C$ denote the pullback of the cospan $A \rightarrow B \leftarrow C$, with the canonical projections $\pi : A \times_{f, B, g} C \rightarrow A$ and $\pi' : A \times_{f, B, g} C \rightarrow C$.

Definition D.2. A double category¹⁰ $\mathbb{D}$ consists of

- A category $\mathbb{D}_0$, which we refer to as the **vertical category** of $\mathbb{D}$.
- A category $\mathbb{D}_1$, equipped with two functors $L, R : \mathbb{D}_1 \rightarrow \mathbb{D}_0$, called the **left and right frame functors**. Given an object $\mathcal{P} \in \mathbb{D}_1$ with $C = L(\mathcal{P})$ and $\mathcal{D} = R(\mathcal{P})$, we say that $\mathcal{P}$ is a **proarrow** (or **horizontal arrow**) from C to $\mathcal{D}$ and write $\mathcal{P} : C \rightarrow \mathcal{D}$. We call morphisms $\phi : \mathcal{P} \rightarrow \mathcal{P}'$ in $\mathbb{D}_1$ **2-cells**. We say a 2-cell ϕ is **globular** if $L(\phi)$ and $R(\phi)$ are identity morphisms in $\mathbb{D}_0$.
- A functor $U : \mathbb{D}_0 \rightarrow \mathbb{D}_1$, which we call the **unit**, such that $L \circ U = R \circ U = 1_{\mathbb{D}_0}$.
- A functor $\odot : \mathbb{D}_1 \times_{R, \mathbb{D}_0, L} \mathbb{D}_1 \rightarrow \mathbb{D}_1$ called **horizontal composition**, such that $L \circ \odot = L \circ \pi$ and $R \circ \odot = R \circ \pi'$, which is weakly associative and weakly unital in the sense that there are coherent, globular left and right unitor and associator isomorphisms [JY20, Definition 12.3.7] denoted λ , ρ , and α , respectively.

Given a double category $\mathbb{D}$ and objects $c, d \in \mathbb{D}_0$, let $\mathbb{D}(c, d)$ denote the subcategory of $\mathbb{D}_1$ whose objects are proarrows $p \in \mathbb{D}_1$ such that $L(p) = c$ and $R(p) = d$, and whose morphisms are globular 2-cells $\phi : p \rightarrow p'$.

Definition D.3. A strong double functor $\mathbb{F} : \mathbb{D} \rightarrow \mathbb{E}$ of double categories consists of

- A functor $\mathbb{F}_0 : \mathbb{D}_0 \rightarrow \mathbb{E}_0$.
- A functor $\mathbb{F}_1 : \mathbb{D}_1 \rightarrow \mathbb{E}_1$ such that $L \circ \mathbb{F}_1 = \mathbb{F}_0 \circ L$ and $R \circ \mathbb{F}_1 = \mathbb{F}_0 \circ R$.
- Globular isomorphisms $\mu : \mathbb{F}_1 \circ \odot \cong \odot \circ \mathbb{F}_1 \times_{\mathbb{F}_0} \mathbb{F}_1$ and $\eta : \mathbb{F}_1 \circ U \cong U \circ \mathbb{F}_0$ which are coherent with respect to the left and right unitors and associators of $\mathbb{D}$ and $\mathbb{E}$ (see [Fer06] for the full definition).

¹⁰By a double category, we mean what is sometimes called a weak or pseudo double category in the literature [Gra19]. Namely we consider horizontal arrow composition to be weakly unital and associative, in contrast to strict double categories

In this paper we will only consider double functors that are strong, and thus omit the word “strong.”

Definition D.4. Let $\mathbb{D}$ be a double category, and let $\sim_i$ be a congruence on $\mathbb{D}_i$ for $i = 0, 1$. If L, R, U , and $\odot$ preserve these congruences, we say that $\sim = (\sim_0, \sim_1)$ is a double congruence on $\mathbb{D}$. Then we can define a double category $\mathbb{D}/\sim$, by $\mathbb{D}_i/\sim_i$ for $i = 0, 1$ and the functors induced by L, R, U and $\odot$.

Lemma D.5. Given a double category $\mathbb{D}$ with a double congruence $\sim$, every double functor $F : \mathbb{D} \rightarrow \mathbb{D}'$ factors uniquely through $\mathbb{D}/\sim$ to induce a functor $(\mathbb{D}/\sim) \rightarrow \mathbb{D}'$.

Definition D.6 (Non-globular morphisms). Let $\mathbf{Prof} := \mathbf{Cat}/2$ and let $\mathbf{Inst}$ be the full subcategory of $\mathbf{Prof}$ on instances. Now define the category $\mathbf{InstPr}$ as follows: its objects are C -instance presentations for some $C \in \mathbf{CatPr}$, and a morphism F from $I \in C\text{-InstPr}$ to $J \in D\text{-InstPr}$ is a morphism of category presentations $|I| \rightarrow |J|$ which sends $*$ to $*$ and sorts of C to sorts of D . In other words, it is given by a morphism of category presentations $F_1 : C \rightarrow D$ and a function $F_{01} : \text{Gen}(I) \rightarrow \text{Term}(J)$ such that $p : c$ implies $F_{01}(p) : F_1(c)$ and for all equations $p.f =_I p'.f'$, $F_{01}(p).F_1(f) \approx_J F_{01}(p').F_1(f')$. Let $\mathbf{InstPr}_{\approx}$ be defined by quotienting by $\approx$, which holds between F and G if and only if $F_1 \approx G_1$ and $F_{01}(p) \approx_J G_{01}(p)$ for all $p \in \text{Gen}(I)$.

Finally, define the category $\mathbf{Curr}$ as follows: its objects are curried profunctor presentations, and a morphism F from $P : C \rightarrow D$ to $P' : C' \rightarrow D'$ consists of category presentation morphisms $F_0 : C \rightarrow C'$ and $F_1 : D \rightarrow D'$, along with a collection of morphisms $F_c : P(c) \rightarrow P'(F_0(c))$ in $\mathbf{InstPr}$ with $(F_c)_1 = F_1$, indexed by the sorts c of C , such that for each function symbol $f : c \rightarrow c'$ in C , the following diagram commutes up to provable equality:

$$\begin{array}{ccc} P(c') & \xrightarrow{F_{c'}} & P'(F_0(c')) \\ P(f) \downarrow & \approx & \downarrow P'(F_0(f)) \\ P(c) & \xrightarrow{F_c} & P'(F_0(c)) \end{array} \quad (5)$$

Notice that (5) still commutes when f is a path, by induction on f ; thus these morphisms can be composed in the natural way. Let $\mathbf{FinCurr}$ denote the full subcategory of finite curried profunctor presentations, i.e. P such that C, D , and $P(c)$ for each c are finite. We say that two curried profunctor presentation morphisms $F, G : P \rightarrow P'$ are **provably equal**, and write $F \approx G$, if $F_0 \approx G_0$, $F_1 \approx G_1$, and $F_c \approx G_c : P(c) \rightarrow P'(F_0(c)) = P'(G_0(c))$ for each $c \in C$.

Definition D.7. Let $L, R : \mathbf{Prof} \rightarrow \mathbf{Cat}$ be the functors defined objectwise by sending a profunctor $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$ to the categories $\mathcal{C}$ and $\mathcal{D}$ respectively. Then the operation $\odot$ of profunctor composition in Definition 3.3 extends to a functor $\odot : \mathbf{Prof} \times_{R, \mathbf{Cat}, L} \mathbf{Prof} \rightarrow \mathbf{Prof}$. Let $U : \mathbf{Cat} \rightarrow \mathbf{Prof}$ be defined on objects by sending $\mathcal{C}$ to the profunctor $\text{pr}_2 : \mathcal{C} \times \mathbf{2} \rightarrow \mathbf{2}$ and on morphisms by sending a functor $F : \mathcal{C} \rightarrow \mathcal{D}$ to the morphism of profunctors $F \times \mathbf{2} : \mathcal{C} \times \mathbf{2} \rightarrow \mathcal{D} \times \mathbf{2}$. Together, the functors L, R, U and $\odot$ form a double category $\mathbb{P}\mathbf{Prof}$ such that $\mathbb{P}\mathbf{Prof}_0 := \mathbf{Cat}$ and $\mathbb{P}\mathbf{Prof}_1 := \mathbf{Prof}$. In other words, the objects are (small) categories, vertical morphisms are functors, horizontal morphisms are profunctors, with horizontal composition given by profunctor composition and the 2-cells are morphisms of profunctors. Note that $\mathbb{P}\mathbf{Prof}(\mathcal{C}, \mathcal{D})$ is the subcategory of $\mathbf{Prof}$ whose objects are profunctors $\mathcal{P} : \mathcal{C} \rightarrow \mathcal{D}$, and whose morphisms $F : \mathcal{P} \rightarrow \mathcal{P}'$ restrict to the identity on $\mathcal{C}$ and $\mathcal{D}$. See [Sch+17, Example 2.12] and [Lor21, Remark 5.1.8] for more details.

Definition D.8. Let $L_{\mathbf{Curr}}, R_{\mathbf{Curr}} : \mathbf{Curr} \rightarrow \mathbf{CatPr}$ denote the left and right frame functors, sending a curried profunctor presentation $P : C \rightarrow D$ to the category presentations C and D respectively.

Lemma D.9. The construction $P \mapsto \langle P \rangle$ of Definition 3.18 extends to a functor $\langle - \rangle : \mathbf{Curr} \rightarrow \mathbf{Prof}$.

Proof. Given curried profunctor presentations $P : C \rightarrowtail D$ and $P' : C' \rightarrowtail D'$ and a morphism $\phi : P \rightarrow P'$ between them, let $\langle \phi \rangle : \langle P \rangle \rightarrow \langle P' \rangle$ be defined as the following functor of categories over $\mathbf{2}$. On objects and morphisms of $\langle C \rangle$ it acts as $\langle \phi_0 \rangle : \langle C \rangle \rightarrow \langle C' \rangle$, while on objects and morphisms of $\langle D \rangle$ it acts as $\langle \phi_1 \rangle : \langle D \rangle \rightarrow \langle D' \rangle$. Finally, given any cross-morphism $[t] \in \langle P(c) \rangle(d) = \langle P \rangle(c, d)$, let $\langle \phi \rangle([t]) := [\phi_c(t)]$. It is straightforward to see that this gives a well defined functor which commutes with the projection functors into $\mathbf{2}$. Functoriality is also immediate from the definition. $\square$

Lemma D.10. The assignment $(P, Q) \mapsto P \otimes Q$ defines a functor $\otimes : \mathbf{Curr} \times_{R_{\mathbf{Curr}}, \mathbf{CatPr}, L_{\mathbf{Curr}}} \mathbf{Curr} \rightarrow \mathbf{Curr}$.

Proof. This is just a generalization of Lemma 3.25 to non-globular morphisms of curried profunctor presentations. Accordingly, the proof is completely analogous. For concreteness, we give the action of $\otimes$ on non-globular morphisms.

Given curried profunctor presentations $P : C \rightarrowtail D$, $P' : C' \rightarrowtail D'$, $Q : D \rightarrowtail E$ and $Q' : D' \rightarrowtail E'$, and given non-globular morphisms $\phi : P \rightarrow P'$ and $\psi : Q \rightarrow Q'$ such that $R_{\mathbf{Curr}}(\phi) = L_{\mathbf{Curr}}(\psi)$, let $\phi \otimes \psi : P \otimes Q \rightarrow P' \otimes Q'$ be defined by $(\phi \otimes \psi)_0 := \phi_0 : C \rightarrow C'$, and $(\phi \otimes \psi)_1 := \psi_1 : E \rightarrow E'$, and for each $c \in \text{Sort}(C)$, let $(\phi \otimes \psi)_c : (P \otimes Q)(c) \rightarrow (P' \otimes Q')(\phi_0(c))$ be defined as in Lemma 3.25 by sending generators $(p \otimes q)$ to $(\phi_c(p) \otimes \psi_d(q))$, where $d = t(p)$. Using this definition, the proof follows the same steps as those of Lemma 3.25. $\square$

Theorem 5.1. There exists a double category $\mathbf{Curr}$ such that the objects are category presentations and the horizontal morphisms are curried profunctor presentations.

Proof. Let us show that the functors $L_{\mathbf{Curr}}$, $R_{\mathbf{Curr}}$, and $\otimes$ form a double category $\mathbf{Curr}$ whose objects are category presentations, with vertical morphisms given by morphisms of category presentations, horizontal morphisms given by curried profunctor presentations and 2-cells given by morphisms of curried profunctor presentations.

We first define a unit functor $U : \mathbf{CatPr} \rightarrow \mathbf{Curr}$. For a category presentation C and a sort $c \in C$, we define $U(C)(c)$ to be the C -instance presentation with one generator $(x_c : c)$ and no equations. Given a function symbol $(f : c \rightarrow c')$ in C , we define $U(C)(f) : U(C)(c') \rightarrow U(C)(c)$ by $(x_{c'} \mapsto x_c.f)$. Given a category presentation morphism $F : C \rightarrow C'$, we define $U(F) : U(C) \rightarrow U(C')$ by $U(F)_0 = U(F)_1 = F$ and, for each $c \in C$, we define $U(F)_c : U(C)(c) \rightarrow U(C')(F_c)$ by $(x_c \mapsto x_{F_c})$. Clearly, we have $L \circ U = R \circ U = 1_{\mathbf{CatPr}}$.

Next, we define the globular associator and unitor isomorphisms $\alpha^{P,Q,R} : (P \otimes Q) \otimes R \cong P \otimes (Q \otimes R)$, $\lambda^P : I(L(P)) \otimes P \cong P$, and $\rho^P : P \otimes I(R(P)) \cong P$, natural in all variables.

For a generator $((p \otimes q) \otimes r) \in \text{Gen}(((P \otimes Q) \otimes R)(c))$, we define $\alpha_c^{P,Q,R}((p \otimes q) \otimes r) := p \otimes (q \otimes r)$. Then for terms $(s : d) \in \text{Term}(P(c))$, $(t : e) \in \text{Term}(Q(d))$, and $(u : f) \in \text{Term}(R(e))$, let $s \equiv p.g$, $Q(g)(t) \equiv q.h$, and $R(h)(u) \equiv r.k$. Then $\alpha_c^{P,Q,R}$ maps $(s \otimes t) \otimes u \equiv (p \otimes Q(g)(t)) \otimes u \equiv (p \otimes q).h \otimes u \equiv ((p \otimes q) \otimes r).k$ to $(p \otimes (q \otimes r)).k$, which is easily shown to be syntactically equal to $s \otimes (t \otimes u)$. Using this fact we can easily verify that $\alpha^{P,Q,R}$ is a curried profunctor presentation morphism, and it is also clearly invertible and globular. This fact also makes naturality in P , Q , and R straightforward to show.

Next, for a generator $p \in \text{Gen}(P(c))$, we define $\lambda_c^P(x_c \otimes p) := p$. Clearly λ_c^P is an invertible C -instance morphism. Given a C -function symbol $f : c \rightarrow c'$, we have $\lambda_c^P(U(P)(f)(x_{c'}) \otimes p) \equiv \lambda_c^P(x_c \otimes P(f)(p)) \equiv P(f)(p) \equiv P(f)(\lambda_{c'}^P(x_{c'} \otimes p))$, so λ^P is an isomorphism, and it is easy to see that it is natural in P and globular.

For a generator $(p : d) \in \text{Gen}(P(c))$, we define $\rho_c^P(p \otimes x_d) \equiv p$. For a D -path $g : d \rightarrow d'$, we have $\rho_c^P(p.g \otimes x_{d'}) \equiv \rho_c^P(p \otimes x_d.g) \equiv p.g$. Just as above, ρ^P is an isomorphism, natural in P , and globular.

The pentagon and triangle identities are trivially satisfied. $\square$

Theorem 5.2. The construction $\llbracket - \rrbracket$ on curried profunctor presentations defines a double functor $\llbracket - \rrbracket : \mathbf{Curr} \rightarrow \mathbf{Prof}$.

Proof. Let $\llbracket - \rrbracket_0 : \mathbf{Curr}_0 \rightarrow \mathbf{Prof}_0$ and $\llbracket - \rrbracket_1 : \mathbf{Curr}_1 \rightarrow \mathbf{Prof}_1$ be given by the semantic functors $\llbracket - \rrbracket : \mathbf{CatPr} \rightarrow \mathbf{Cat}$ and $\llbracket - \rrbracket : \mathbf{Curr} \rightarrow \mathbf{Prof}$. Notice that indeed $L \circ \llbracket - \rrbracket_1 = \llbracket - \rrbracket_0 \circ L_{\mathbf{Curr}}$ and $R \circ \llbracket - \rrbracket_1 = \llbracket - \rrbracket_0 \circ R_{\mathbf{Curr}}$ as required. The desired globular isomorphism $\mu : \odot \circ \llbracket - \rrbracket_1 \times_{\llbracket - \rrbracket_0} \llbracket - \rrbracket_1 \cong \llbracket - \rrbracket_1 \circ \otimes$ is essentially given by the isomorphism of Theorem 3.26, generalized to non-globular morphisms of curried profunctor presentations. We now describe this more precisely.

We are looking for a natural isomorphism $\mu : \llbracket - \rrbracket \circ \llbracket - \rrbracket \Rightarrow \llbracket - \otimes - \rrbracket$ between functors of the type $\mathbf{Curr} \times_{R_{\mathbf{Curr}}, \text{CatPr}, L_{\mathbf{Curr}}} \mathbf{Curr} \rightarrow \mathbf{Prof}$. This means that we must give a component isomorphism $\mu^{P,Q} : \llbracket P \rrbracket \odot \llbracket Q \rrbracket \Rightarrow \llbracket P \otimes Q \rrbracket$ for each pair of composable curried profunctor presentations P and Q . This is already provided by Theorem 3.26. The only difference, then, is that in order to establish naturality we must now consider curried profunctor presentations $P : C \rightrightarrows D$, $P' : C' \rightrightarrows D'$, $Q : D \rightrightarrows E$ and $Q' : D' \rightrightarrows E'$, together with arbitrary morphisms $\phi : P \rightarrow P'$ and $\psi : Q \rightarrow Q'$, and show that $(\llbracket \phi \otimes \psi \rrbracket) \circ \mu^{P,Q} = \mu^{P',Q'} \circ (\llbracket \phi \rrbracket \odot \llbracket \psi \rrbracket)$ using the functorial action of $\otimes$ on non-globular morphisms given in Lemma D.10. This is equally immediate as the verification in the globular case. To clarify the role of the non-globular morphisms we repeat the verification expanding, without loss of generality, $s \equiv p$ and $t \equiv q.h$ for some $(p : d) \in \text{Gen}(P(c))$, $(q : e') \in \text{Gen}(Q(d))$ and E -path $h : e' \rightarrow e$. We then have that

$$\begin{aligned} (\llbracket \phi \otimes \psi \rrbracket \circ \mu^{P,Q})(\llbracket [p], [q.h] \rrbracket) &= \llbracket \phi \otimes \psi \rrbracket(\llbracket p \otimes q.h \rrbracket) \\ &= [\phi_c(p) \otimes \psi_d(q). \psi_1(h)] \\ &= \mu^{P',Q'}(\llbracket [\phi_c(p)], [\psi_d(q). \psi_1(h)] \rrbracket) \\ &= (\mu^{P',Q'} \circ (\llbracket \phi \rrbracket \odot \llbracket \psi \rrbracket))(\llbracket [p], [q.h] \rrbracket). \end{aligned}$$

Now we define a globular isomorphism $\varepsilon^C : \llbracket U(C) \rrbracket \rightarrow \text{Hom}_{\llbracket C \rrbracket}$ as follows. Given a C -path $f : c \rightarrow c'$, let $\varepsilon_c^C([x_c.f]) = [f]$. To see that this is well-defined, we must show that for $f, f' : c \rightarrow c'$,

$$(x_c.f \approx_{U(C)(c)} x_c.f') \iff (f \approx_C f') \quad (6)$$

For the $(\Rightarrow)$ direction, we observe that $\approx_C$ is an equivalence relation, includes (f, f') for every $(x_c.f = x_c.f') \in U(C)(c)_E = \emptyset$, includes C -equations, and is closed under right-composition with a function symbol. For the $(\Leftarrow)$ direction, we simply observe that $\approx_{|U(C)(c)|}$ specializes to both $\approx_C$ and $\approx_{U(C)(c)}$ and respects left-composition with x_c .

It is easily observed that ε_c^C is a globular morphism of profunctors, and that it is natural in c and C .

Finally, we need to check that μ and ε respect α , λ , and ρ . However, this is straightforward. $\square$

Proposition 5.3. The congruences $\approx$ on $\mathbf{Curr}$ and $\mathbf{CatPres}$ comprise a double congruence on $\mathbf{Curr}$. Define $\mathbf{Curr}_{\approx}$ by quotienting $\mathbf{Curr}$ by $\approx$, as in Def. D.4. The semantics double functor $\llbracket - \rrbracket$ induces a double equivalence $\llbracket - \rrbracket : \mathbf{Curr}_{\approx} \rightarrow \mathbf{Prof}$.

Proof. The semantics double functor $\llbracket - \rrbracket$ is invariant with respect to provable equality, so by Lemma D.5 there is an induced double functor $\llbracket - \rrbracket : \mathbf{Curr}/\approx \rightarrow \mathbf{Prof}$. To show that this strong double functor is an equivalence, [Shu08, Theorem 7.8] tells us that we can show that it is full, faithful, and essentially surjective in the sense of [Shu08, Def. 3.3, 3.6]. This is straightforward. $\square$

Reverse Faà di Bruno’s Formula for Cartesian Reverse Differential Categories

Aaron Biggin*

Jean-Simon Pacaud Lemay†

Macquarie University
Sydney, New South Wales, Australia

aaron.biggin@students.mq.edu.au

js.lemay@mq.edu.au

Reverse differentiation is an essential operation for automatic differentiation. Cartesian reverse differential categories axiomatize reverse differentiation in a categorical framework, where one of the primary axioms is the reverse chain rule, which is the formula that expresses the reverse derivative of a composition. Here, we present the reverse differential analogue of Faà di Bruno’s Formula, which gives a higher-order reverse chain rule in a Cartesian reverse differential category. To properly do so, we also define partial reverse derivatives and higher-order reverse derivatives in a Cartesian reverse differential category.

1 Introduction

There are two types of derivative operations used in automatic differentiation: **forward differentiation** and the **reverse differentiation** (also referred to as forward mode and reverse mode differentiation respectively). For a smooth function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, its forward derivative is the usual total derivative, which is of type $D[F] : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ and can be defined using the Jacobian of F . On the other hand, its reverse derivative is of type $R[F] : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, which can instead be defined in terms of the transpose of the Jacobian of F . As such, the forward derivative and the reverse derivative are the transposes of each other. Forward differentiation is more efficient when the dimension of the output is much larger than the dimension of the input, $n \leq m$. In comparison, reverse differentiation is more efficient when the dimension of the input is much larger than the dimension of the output, $n \geq m$. Since many optimization algorithms, especially those used in machine learning, deal with large input data sets that are characterized by functions of type $\mathbb{R}^n \rightarrow \mathbb{R}$ where n is quite large, it is reverse mode differentiation which is often more practical in these setting. Motivated by this, there has been significant interest in studying the categorical foundations of reverse differentiation [2, 5, 7, 8, 13, 15], which fits in the larger program of providing the categorical foundations for machine learning [6, 9, 10, 12, 14, 16].

Cartesian differential categories [1] provide the categorical foundations of forward differentiation, as well as the categorical semantics of the differential λ -calculus. In particular, a Cartesian differential category comes equipped with a **forward differential combinator** (Sec 3), which is an operator that captures forward differentiation of maps. The axioms of a forward differential combinator are analogues of the fundamental identities of the total derivative from differential calculus, such as the famous chain rule for the forward derivative of a composition. On the other hand, Cartesian reverse differential categories provide the categorical foundations of reverse differentiation. This time, a Cartesian reverse differential category comes equipped with a **reverse differential combinator** (Def 2.1), which is an operator that

*For work on this research, this author was funded by a Summer Vacation Research Scholarship from Macquarie University.

†This material is based upon work supported by the AFOSR under award number FA9550-24-1-0008. This author is also funded by an ARC DECRA (DE230100303).

captures reverse differentiation of maps. The axioms of the reverse differential combinator are the reverse differential counterparts of the axioms of the forward differential combinator. In particular, one of the axioms is the reverse chain rule for the reverse derivative of a composition. Every reverse differential combinator induces a forward differential combinator as well as a transpose operation, called a **contextual linear dagger** (Sec 3), such that the reverse derivative and forward derivative are transposes of one another [2, Thm 42]. As such, every Cartesian reverse differential category is also a Cartesian differential category [2, Thm 16]. Therefore, there is interest in studying and developing the reverse differential counterparts of well-known forward differential concepts in a Cartesian reverse differential category.

Famously, **Faà di Bruno's Formula** provides a higher-order chain rule for the formula of a higher-order forward derivative of a composition. Faà di Bruno's Formula also holds in a Cartesian differential category [11, Lemma 3.4]. The main objective of this paper is to provide the reverse differential version of Faà di Bruno's Formula, giving a higher-order reverse chain rule. As we will see, the reverse Faà di Bruno's Formula surprisingly involves both reverse derivatives and forward derivatives. To properly give the reverse Faà di Bruno's Formula, we will first need to develop the appropriate notions of **partial reverse derivatives** (Def 2.3) and **higher-order reverse derivatives** (Def 4.1) in a Cartesian reverse differential category, in such a way that they are the transposes of their respective forward derivative counterparts. So, we will show that the transpose of the partial forward derivative is indeed the partial reverse derivative (Lemma 3.5). However, to also get that the transpose of the higher order forward derivative is the higher order reverse derivative, we introduce an extra compatibility relation between the reverse differential combinator and the forward differential combinator, which we call the **stable rule** (Def 4.2). This additional rule is not very imposing and quite natural, and holds in prominent examples of Cartesian reverse differential categories (Ex 4.3). Then, in the same way that the reverse chain rule can be computed by taking the transpose of the chain rule, by taking the transpose of Faà di Bruno's Formula, we can compute the reverse Faà di Bruno's Formula (Thm 4.5).

To the best of the authors' knowledge, this is the first expression of Faà di Bruno's Formula for reverse mode differentiation. The reverse Faà di Bruno's Formula will hopefully help improve calculations in (categorical) models for automatic differentiation and machine learning. Moreover, an important application of Faà di Bruno's Formula for Cartesian differential categories is that it is a crucial formula in the construction of *cofree* Cartesian differential categories [4, 11]. Therefore, in future work, it would be interesting to understand if the reverse Faà di Bruno's Formula can be used somehow to construct *cofree* Cartesian reverse differential categories.

Conventions: In an arbitrary category $\mathbb{X}$, objects will be denoted by capital letters A, B, C , etc. and maps by miniscule letters f, g, h , etc. We denote hom-sets by $\mathbb{X}(A, B)$, maps as arrows $f : A \rightarrow B$, identity maps as $1_A : A \rightarrow A$, and we use the classical notation for composition $\circ$, as opposed to diagrammatic order that was used in other papers on Cartesian (reverse) differential categories, such as in [1, 2]. For a category with finite products, we denote the product by $\times$, the projection maps by $\pi_j : A_1 \times \dots \times A_n \rightarrow A_j$, and the pairing operation as $\langle -, \dots, - \rangle$.

2 Reverse Differentiation

In this section, we review Cartesian reverse differential categories and, in particular, also develop the notion of *partial* reverse derivatives. Partial reverse differentiation is very practical and provides a helpful new perspective on the axioms of reverse differentiation and, as we will see in the next section, defining forward differentiation and linear transpose operation from reverse differentiation.

Here, following what was done for forward differentiation in [11], we will introduce the definition of

a Cartesian reverse differential category relative to a fixed commutative semiring k , slightly generalising the original definition from [2]. As such, in this relative setting, the underlying structure of a Cartesian reverse differential category is that of a *Cartesian left k -linear category* [11, Sec 2.1], which can be described as a category with finite products which is *skew-enriched* over the category of k -modules and k -linear maps between them [11]. Essentially, this means that each hom-set is a k -module, so we have zero maps, can take the sum of maps, and can scalar multiply maps by elements of k , but also allow for maps which do not preserve zeroes, sums, or scalar multiplication. Maps which do preserve the module structure are called *k -linear maps*. Explicitly, a **left k -linear category** is a category $\mathbb{X}$ such that each hom-set $\mathbb{X}(A, B)$ is a k -module with scalar multiplication $\cdot : k \times \mathbb{X}(A, B) \rightarrow \mathbb{X}(A, B)$, addition $+$: $\mathbb{X}(A, B) \times \mathbb{X}(A, B) \rightarrow \mathbb{X}(A, B)$, and zero $0 \in \mathbb{X}(A, B)$; and such that pre-composition preserves the k -linear structure: $(s \cdot f + t \cdot g) \circ x = s \cdot (f \circ x) + t \cdot (g \circ x)$. A map $f : A \rightarrow B$ is said to be **k -linear** if post-composition by f preserves the k -linear structure: $f \circ (s \cdot x + t \cdot y) = s \cdot (f \circ x) + t \cdot (f \circ y)$. A **Cartesian left k -linear category** is a left k -linear category $\mathbb{X}$ such that $\mathbb{X}$ has finite products and all projection maps π_j are k -linear. We note that when taking $k = \mathbb{N}$, the semiring of natural numbers, (Cartesian) left $\mathbb{N}$ -linear categories and their $\mathbb{N}$ -linear maps are the same thing as (Cartesian) left additive categories and their additive maps from [1, Def 1.1.1 & 1.2.1].

A Cartesian reverse differential category is a Cartesian k -linear category which comes equipped with a *reverse differential combinator*, which is an operator that sends maps to their reverse derivative. In this relative to k setting, the reverse differential combinator still satisfies essentially the same seven axioms as in [2, Def 13], but where we upgrade the first two [RD.1] and [RD.2] from simple additivity to k -linearity in the obvious way (so [RD.3] to [RD.7] are exactly the same). So, in particular, when taking $k = \mathbb{N}$, a Cartesian $\mathbb{N}$ -reverse differential category is precisely the same thing as the original definition of a Cartesian reverse differential category from [2, Def 13].

Definition 2.1 [2, Def 13] A **Cartesian k -reverse differential category** is a Cartesian left k -linear category that comes equipped with a **reverse differential combinator** R , that is, a family of functions $R : \mathbb{X}(A, B) \rightarrow \mathbb{X}(A \times B, A)$, satisfying the seven axioms [RD.1] to [RD.7] described below. For a map $f : A \rightarrow B$, the map $R[f] : A \times B \rightarrow A$ is called the **reverse derivative** of f .

Before we review the axioms of the reverse differential combinator, it may be helpful to first quickly review the canonical example of a Cartesian reverse differential category:

Example 2.2 Let $\mathbb{R}$ be the set of real numbers. Define **SMOOTH** as the category whose objects are the Euclidean real vector spaces $\mathbb{R}^n$ and whose maps are real smooth functions $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ between them. **SMOOTH** is a Cartesian $\mathbb{R}$ -reverse differential category where the reverse differential combinator is defined using the transpose of the Jacobian, see [2, Ex 14.2] and [13, Ex 4.2]. Explicitly, for a smooth function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (where recall that F is in fact a tuple $F = \langle f_1, \dots, f_m \rangle$ of smooth functions $f_j : \mathbb{R}^n \rightarrow \mathbb{R}$), its reverse derivative $R[F] : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is defined as follows:

$$R[F](\vec{x}, \vec{y}) := \left(\sum_{j=1}^m \frac{\partial f_j}{\partial x_1}(\vec{x}) y_j, \dots, \sum_{j=1}^m \frac{\partial f_j}{\partial x_n}(\vec{x}) y_j \right)$$

Other examples of Cartesian reverse differential categories can be found in [2, 5], which include any dagger category with dagger biproducts [2, Ex 19], polynomials over a fixed semiring [2, Ex 14.1], and the coKleisli category of a monoidal reverse differential category [5, Thm 55].

So what are the seven axioms [RD.1] to [RD.7] of a reverse differential combinator? Simply writing down the equations of [RD.1] to [RD.7] as in [2, Def 13] can be a bit non-intuitive to parse at first glance.

Especially the last two axioms [RD.6] and [RD.7] are a bit complicated to write down and somewhat intimidating at first. To aid in readability, we will use a term calculus for reverse differentiation inspired by the one for forward differentiation [1, Sec 4]. In this term calculus, the reverse derivative $R[f](a, b)$ will be written as:

$$\frac{rf(x)}{rx}(a) \cdot b$$

We leave formalising this term calculus, including proving soundness and completeness, for future work. So, using this term calculus, the first five axioms of the reverse differential combinator are:

[RD.1] The reverse differential combinator is a k -linear morphism:

$$\frac{r(s \cdot f(x) + t \cdot g(x))}{rx}(a) \cdot b = s \cdot \frac{rf(x)}{rx}(a) \cdot b + t \cdot \frac{rg(x)}{rx}(a) \cdot b$$

[RD.2] Reverse derivatives are k -linear in their second argument:

$$\frac{rf(x)}{rx}(a) \cdot (s \cdot b + t \cdot c) = s \cdot \frac{rf(x)}{rx}(a) \cdot b + t \cdot \frac{rf(x)}{rx}(a) \cdot c$$

[RD.3] The reverse derivative of identities and projections are:

$$\frac{rx}{rx}(a) \cdot b = b \quad \frac{rx_j}{r(x_1, \dots, x_n)}(a_0, \dots, a_n) \cdot b = \underbrace{(0, \dots, 0, b, 0, \dots, 0)}_{j\text{th-component}}$$

[RD.4] The reverse derivative of a tuple is the sum of the reverse derivatives of the components of the tuple:

$$\frac{r\langle f_1(x), \dots, f_n(x) \rangle}{rx}(a) \cdot (b_1, \dots, b_n) = \sum_{j=1}^n \frac{rf_j(x)}{rx}(a) \cdot b_j$$

[RD.5] The reverse chain rule formula for the reverse derivative of a composition:

$$\frac{rg(f(x))}{rx}(a) \cdot b = \frac{rf(x)}{rx}(a) \cdot \left(\frac{rg(y)}{ry}(f(a)) \cdot b \right)$$

To help write the last two axioms [RD.6] and [RD.7], it will be useful to first introduce the concept of *partial* reverse derivatives. So, given a map of type $f : A_1 \times \dots \times A_n \rightarrow B$, we'd like to be able to take the reverse derivative of f with respect to the component A_j while keeping the rest constant. To do so, consider first the total reverse derivative of f , which is of type $R[f] : A_1 \times \dots \times A_n \times B \rightarrow A_1 \times \dots \times A_n$. By the universal property of the product, we know that in this case, $R[f]$ is actually a tuple of maps of type $A_1 \times \dots \times A_n \times B \rightarrow A_j$ which are, of course, defined by post-composing $R[f]$ with the respective projections. We interpret these maps as the partial reverse derivatives of f .

Definition 2.3 *In a Cartesian k -reverse differential category, for a map $f : A_1 \times \dots \times A_n \rightarrow B$, its j -th partial reverse derivative is the map $R_j[f] : A_1 \times \dots \times A_n \times B \rightarrow A_j$ defined as the following composite:*

$$R_j[f] : A_1 \times \dots \times A_n \times B \xrightarrow{R[f]} A_1 \times \dots \times A_n \xrightarrow{\pi_j} A_j \quad (1)$$

In the term calculus, we write partial reverse derivatives as follows:

$$\frac{rf(a_1, \dots, a_{j-1}, x_j, a_{j+1}, \dots, a_n)}{rx_j}(a_j) \cdot b := \pi_j \left(\frac{rf(x_1, \dots, x_n)}{r(x_1, \dots, x_n)}(a_1, \dots, a_n) \cdot b \right)$$

Example 2.4 For a smooth function $F = \langle f_1, \dots, f_m \rangle : \mathbb{R}^n \rightarrow \mathbb{R}^m$, its j -th partial reverse derivative $R_j[F] : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ is the sum of the partial derivatives in the x_j variable:

$$R_j[F](\vec{x}, \vec{y}) = \sum_{k=1}^m \frac{\partial f_k}{\partial x_j}(\vec{x}) y_k \quad (2)$$

Recall that the reverse derivative of a smooth function was the tuple of the partial derivatives. The same idea holds in a Cartesian reverse differential category, which is, of course, immediate from the universal property of the product. We note that this is the reverse differential analogue of the fact that the total forward derivative is equal to the sum of the partial forward derivatives [11, Lemma 2.8.(i)].

Lemma 2.5 In a Cartesian k -reverse differential category, for a map $f : A_1 \times \dots \times A_n \rightarrow B$, we have that $R[f] = \langle R_1[f], \dots, R_n[f] \rangle$, which in the term calculus is expressed as:

$$\frac{rf(x_1, \dots, x_n)}{r(x_1, \dots, x_n)}(a_1, \dots, a_n) \cdot b = \left\langle \frac{rf(x_1, a_2, \dots, a_n)}{rx_1}(a_1) \cdot b, \dots, \frac{rf(a_1, \dots, x_n)}{rx_n}(a_n) \cdot b \right\rangle \quad (3)$$

Now that we have partial reverse derivatives, we can smoothly write down the remaining two axioms:

[RD.6] Involution of the *linear transpose*:

$$\frac{\frac{rf(x)}{rx}(a) \cdot u}{\frac{ru}{rv}(0) \cdot v} = \frac{rf(x)}{rx}(a) \cdot b$$

[RD.7] Symmetry of the mixed partial *forward* derivatives:

$$\frac{\frac{\frac{rf(x)}{rx}(y) \cdot u}{ru}(0) \cdot b}{ry}(a) \cdot v = \frac{\frac{\frac{rf(x)}{rx}(y) \cdot u}{ru}(0) \cdot c}{ry}(a) \cdot v$$

While these axioms may still look somewhat strange initially, they will make more sense when considering the relationship between reverse differentiation and forward differentiation – which we discuss in the next section below. For a more in-depth discussion on the seven reverse differential combinator axioms (and Cartesian reverse differential categories in general), we invite the reader to see [2, 5].

Another way of describing partial reverse differentiation is as reverse differentiation in context. From this point of view, partial reverse differentiation is an actual reverse differential combinator for simple slice categories. Recall that for a category $\mathbb{X}$ with finite products, for each object $C \in \mathbb{X}$, the **simple slice category** over C is the category $\mathbb{X}[C]$ whose objects are the same as $\mathbb{X}$ and whose homsets are $\mathbb{X}[C](A, B) = \mathbb{X}(C \times A, B)$. Composition of $f : C \times A \rightarrow B$ and $g : C \times B \rightarrow D$ is $g \circ \langle \pi_1, f \rangle$, which is written in the term calculus as $g(c, f(c, x))$, and the identity map is the projection $\pi_2 : C \times A \rightarrow A$. $\mathbb{X}[C]$ also has finite products, and if $\mathbb{X}$ is a Cartesian left k -linear category, then $\mathbb{X}[C]$ will also be a Cartesian k -linear category with the same k -linear structure as $\mathbb{X}$. Showing that if $\mathbb{X}$ is a Cartesian k -reverse differential category then so is $\mathbb{X}[C]$ amounts to showing that partial reverse differentiation satisfies the seven reverse differential combinator axioms in *context*.

Lemma 2.6 In a Cartesian k -reverse differential category, the following equalities hold:

$$[\mathbf{RD.1}] \quad \frac{r(s \cdot f(c_1, x, c_2) + t \cdot g(c_1, x, c_2))}{rx}(a) \cdot b = s \cdot \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b + t \cdot \frac{rg(c_1, x, c_2)}{rx}(a) \cdot b$$

$$[\mathbf{RD.2}] \quad \frac{rf(c_1, x, c_2)}{rx}(a) \cdot (s \cdot b + t \cdot c) = s \cdot \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b + t \cdot \frac{rf(c_1, x, c_2)}{rx}(a) \cdot c$$

$$[\mathbf{RD.3}] \quad \frac{rx_j}{rx_j}(a) \cdot b = b \text{ and } \frac{rx_i}{rx_i}(a) \cdot b = 0 \text{ if } i \neq j;$$

$$[\mathbf{RD.4}] \quad \frac{r\langle f_1(c_1, x, c_2), \dots, f_n(c_1, x, c_2) \rangle}{rx}(a) \cdot (b_0, \dots, b_n) = \sum_{j=1}^n \frac{rf_j(c_1, x, c_2)}{rx}(a) \cdot b_j$$

$$[\mathbf{RD.5}] \quad \frac{rg(c_1, f(c_1, x, c_2), c_2)}{rx}(a) \cdot b = \frac{rf(c_1, x, c_2)}{rx}(a) \cdot \left(\frac{rg(c_1, y, c_2)}{ry}(f(c_1, a, c_2)) \cdot b \right)$$

$$[\mathbf{RD.6}] \quad \frac{\frac{rf(c_1, x, c_2)}{rx}(a) \cdot u}{ru}(0) \cdot v = \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b$$

$$[\mathbf{RD.7}] \quad \frac{\frac{\frac{rf(c_1, x, c_2)}{rx}(y) \cdot u}{ru}(0) \cdot b}{ry}(a) \cdot v = \frac{\frac{\frac{rf(c_1, x, c_2)}{rx}(y) \cdot u}{ru}(0) \cdot c}{ry}(a) \cdot v$$

PROOF: To prove these, we first need to compute the following useful identity:

$$\begin{aligned} & \frac{r\langle c_1, f(c_1, x, c_2), c_2 \rangle}{rx}(a) \cdot (b_1, b_2, b_3) \stackrel{\text{Def.}}{=} \pi_2 \left(\frac{r\langle v, f(v, x, u), u \rangle}{r(v, x, u)}(c_1, a, c_2) \cdot (b_1, b_2, b_3) \right) \\ & \stackrel{[\mathbf{R.4}]}{=} \pi_2 \left(\frac{rv}{r(v, x, u)}(c_1, a, c_2) \cdot b_1 + \frac{rf(v, x, u)}{r(v, x, u)}(c_1, a, c_2) \cdot b_2 + \frac{ru}{r(v, x, u)}(c_1, a, c_2) \cdot b_3 \right) \\ & \stackrel{[\mathbf{R.3}] + (3)}{=} \pi_2 \left((b_1, 0, 0) + \left(\frac{rf(v, a, c_2)}{rv}(c_1) \cdot b_2, \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b_2, \frac{rf(c_1, a, u)}{ru}(c_2) \cdot b_2 \right) + (0, 0, b_3) \right) \\ & \stackrel{\pi_j \text{ is } k\text{-lin.}}{=} \pi_2(b_1, 0, 0) + \pi_2 \left(\frac{rf(v, a, c_2)}{rv}(c_1) \cdot b_2, \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b_2, \frac{rf(c_1, a, u)}{ru}(c_2) \cdot b_2 \right) + \pi_2(0, 0, b_3) \\ & = 0 + \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b_2 + 0 = \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b_2 \end{aligned}$$

So we have that:

$$\frac{r\langle c_1, f(c_1, x, c_2), c_2 \rangle}{rx}(a) \cdot (b_1, b_2, b_3) = \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b_2 \quad (4)$$

Using this and the corresponding axiom **[RD.#]**, we can prove the corresponding partial reverse version. Let's prove the reverse chain rule in context:

$$\begin{aligned} & \frac{rg(c_1, f(c_1, x, c_2), c_2)}{rx}(a) \cdot b \stackrel{\text{Def.}}{=} \pi_2 \left(\frac{rg(v, f(v, x, u), u)}{r(v, x, u)}(c_1, a, c_2) \cdot b \right) \\ & \stackrel{[\mathbf{R.5}]}{=} \pi_2 \left(\frac{r\langle v, f(v, x, u), u \rangle}{r(v, x, u)}(c_1, a, c_2) \cdot \left(\frac{rg(w, y, z)}{r(w, y, z)}(c_1, f(c_1, a, c_2), c_2) \cdot b \right) \right) \\ & \stackrel{\text{Def.}}{=} \frac{r\langle c_1, f(c_1, x, c_2), c_2 \rangle}{rx}(a) \cdot \left(\frac{rg(w, y, z)}{r(w, y, z)}(c_1, f(c_1, a, c_2), c_2) \cdot b \right) \end{aligned}$$

$$\begin{aligned}
& \stackrel{(3)}{=} \frac{r\langle c_1, f(c_1, x, c_2), c_2 \rangle}{rx}(a) \cdot \left(\frac{rg(w, f(c_1, a, c_2), c_2)}{rw}(c_1) \cdot b, \frac{rg(c_1, y, c_2)}{ry}(f(c_1, a, c_2)) \cdot b, \frac{rg(c_1, f(c_1, a, c_2), z)}{rz}(c_2) \cdot b \right) \\
& \stackrel{(4)}{=} \frac{rf(c_1, x, c_2)}{rx}(a) \cdot \frac{rg(c_1, y, c_2)}{ry}(f(c_1, a, c_2)) \cdot b
\end{aligned}$$

The other identities are shown in a similar fashion. $\square$

Then, it follows from the above lemma that the simple slice categories of a Cartesian reverse differential category are again Cartesian reverse differential categories. This is the reverse differentiation analogue of [1, Cor 4.5.2].

Corollary 2.7 *Let $\mathbb{X}$ be a Cartesian k -reverse differential category. Then for each object $C \in \mathbb{X}$, the simple slice category $\mathbb{X}[C]$ is a Cartesian k -reverse differential category whose reverse differential combinator R^C sends a map $f : C \times A \rightarrow B$ to the map $R^C[f] : C \times A \times B \rightarrow A$ defined as the following composite:*

$$R^C[f] : C \times A \times B \xrightarrow{R[f]} C \times A \xrightarrow{\pi_2} A \quad (5)$$

PROOF: The necessary reverse differential combinator axioms with respect to composition in the simple slice category are precisely the identities from Lemma 2.6 with context only on the left. $\square$

3 Forward Differentiation and Linear Transpose

The fundamental theorem for Cartesian reverse differential categories is that they are precisely the same as Cartesian *forward* differential categories equipped with a suitable notion of transpose operation, called a *contextual linear dagger*. In particular, this says we obtain forward differentiation and a transpose operation from reverse differentiation, and vice-versa.

Theorem 3.1 [2, Thm 42] *A Cartesian k -reverse differential category is precisely a Cartesian k -differential category with a contextual k -linear dagger.*

PROOF: The proof in the relative to k setting is essentially the same as was done in [2]. $\square$

We will not thoroughly review Cartesian differential categories and contextual linear daggers here. For a more in-depth introduction to Cartesian forward differential categories, we invite the reader to see [1, 11, 13], and for an in-depth introduction to contextual linear daggers, we invite the reader to see [2, 5]. Instead, we will focus on revisiting how to build a forward differential combinator and contextual linear dagger from a reverse differential combinator. We will also see how these constructions can be nicely expressed using partial reverse derivatives.

Briefly, a **Cartesian k -differential category** [11, Sec 2.2] is a Cartesian k -linear category which comes equipped with a **forward differential combinator**, which is an operator which sends maps $f : A \rightarrow B$ to their **forward derivative** $D[f] : A \times A \rightarrow B$. The differential combinator axioms are analogues of the well-known identities of the total derivative from differential calculus, such as the chain rule. As previously mentioned, Cartesian differential categories have a very practical term calculus [1, Sec 4]. So, we write the forward derivative as:

$$D[f](a, b) := \frac{df(x)}{dx}(a) \cdot b$$

In particular, one the axioms for the forward differential combinator is **[CD.5]** the chain rule for differentiating a composition of maps, which in the term calculus is expressed as:

$$\frac{dg(f(x))}{dx}(a) \cdot b = \frac{dg(y)}{dy}(f(a)) \cdot \left(\frac{df(x)}{dx}(a) \cdot b \right) \quad (6)$$

Let's now revisit how to go from reverse differentiation to forward differentiation by rewriting the construction of [2, Thm 16] using partial reverse derivatives. Consider a map $f : A \rightarrow B$. We need to build a map of type $A \times A \rightarrow B$. We can first take its reverse derivative to get $R[f] : A \times B \rightarrow A$, and then by taking the partial reverse derivative with respect to B , we get a map of type $R_2[R[f]] : A \times B \times A \rightarrow B$. Then inserting zero into the B argument of the domain gives us a map of type $A \times A \rightarrow B$ as desired. Explicitly, for a map $f : A \rightarrow B$, its forward derivative is defined as follows:

$$\frac{df(x)}{dx}(a) \cdot b := \frac{r \frac{rf(x)}{rx}(a) \cdot y}{ry}(0) \cdot b \quad (7)$$

Then **[RD.7]** is precisely expressing the axiom **[CD.7]** of the forward differential combinator (which amounts to the symmetry of the mixed partial forward derivatives) using only reverse differentiation.

Example 3.2 In SMOOTH, the forward differential combinator is the usual total derivative from calculus, which is instead defined using the Jacobian [2, Ex 5.3]. Explicitly, for a smooth function $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ (where recall that F is in fact a tuple $F = \langle f_1, \dots, f_m \rangle$ of smooth functions $f_j : \mathbb{R}^n \rightarrow \mathbb{R}$), its forward derivative $D[F] : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ is defined as follows:

$$D[F](\vec{x}, \vec{y}) := \left(\sum_{i=1}^n \frac{\partial f_1}{\partial x_i}(\vec{x}) y_i, \dots, \sum_{i=1}^n \frac{\partial f_m}{\partial x_i}(\vec{x}) y_i \right)$$

Before defining the transpose operation, we first need to quickly discuss *partial* forward derivatives and *differential linearity*. Starting with partial forward derivatives: given a map $f : A_0 \times \dots \times A_n \rightarrow B$, we'd like to take the forward derivative of f with respect to the component A_j while keeping the rest constant. In differential calculus, partial derivatives are obtained by inserting zeroes in the appropriate vector argument of the total derivative. The same idea holds in a Cartesian differential category. Then the **j -th partial forward derivative** [11, Def 2.7] of f is the map $D_j[f] : A_1 \times \dots \times A_n \times A_j \rightarrow B$ written in term calculus on the left below and is defined on the right as follows:

$$\frac{df(a_1, \dots, a_{j-1}, x_j, a_{j+1}, \dots, a_n)}{dx_j}(a_j) \cdot b := \frac{df(x_1, \dots, x_n)}{d(x_1, \dots, x_n)}(a_1, \dots, a_n) \cdot (0, \dots, 0, b, 0, \dots, 0) \quad (8)$$

A map $f : A_1 \times \dots \times A_n \rightarrow B$ is said to be **differential linear (D-linear)** [11, Def 2.6] in A_j if when taking its j -th partial derivative, the following equality holds:

$$\frac{df(a_1, \dots, a_{j-1}, x_j, a_{j+1}, \dots, a_n)}{dx_j}(a_j) \cdot b = f(a_1, \dots, a_{j-1}, b, a_{j+1}, \dots, a_n)$$

Properties of differential linear maps can be found in [3, Lemma 2.6], such as the fact that they are closed under composition, k -linear structure, etc. In particular, if f is D-linear in its j -th variable, then it is also k -linear in its j -th variable [3, Lemma 2.6.(i)] (though the converse is not necessarily true in an arbitrary Cartesian differential category). Moreover, the axiom **[CD.6]** of the forward differential combinator is precisely the statement that forward derivatives are D-linear in their second argument.

A **contextual linear dagger** [2, Def 39] is an involutive and contravariant operation on maps with a D-linear argument, which swaps the codomain with said D-linear argument. Here we will write down the contextual linear dagger with context both on the left and right, which is equivalent to simply having context on the left as in [2]. So for a map $f : C_1 \times A \times C_2 \rightarrow B$ which is D-linear in A , the **D-linear transpose in A** of f is the map the map $f^\dagger[C_1 \times \_ \times C_2] : C_1 \times B \times C_2 \rightarrow A$ defined as the partial reverse derivative in A of f evaluated at 0:

$$f^\dagger[C_1 \times \_ \times C_2](c_1, b, c_2) := \frac{rf(c_1, x, c_2)}{rx}(0) \cdot b \quad (9)$$

Now $f^\dagger[C_1 \times \_ \times C_2] : C_1 \times B \times C_2 \rightarrow A$ is D-linear in B [2, Cor 36] and this operation is involutive [2, Lemma 35], that is, the D-linear transpose of $f^\dagger[C_1 \times \_ \times C_2]$ is f . Other properties of the contextual linear dagger can be found in [2, 5]. In particular, it will be useful for the reverse Faà di Bruno's Formula to recall the fact that the contextual linear dagger is contravariant with respect to composition in context.

Lemma 3.3 *In a Cartesian k -reverse differential category, if $f : C_1 \times A \times C_2 \rightarrow B$ is D-linear in A and $g : C_1 \times B \times C_2 \rightarrow E$ is D-linear in B , then $(g \circ \langle \pi_1, f, \pi_3 \rangle)^\dagger[C_1 \times \_ \times C_2] = f^\dagger[C_1 \times \_ \times C_2] \circ \langle \pi_1, g^\dagger[C_1 \times \_ \times C_2], \pi_3 \rangle$, which in the term logic is explicitly expressed as:*

$$\frac{rg(c_1, f(c_1, x, c_2), c_2)}{rx}(0) \cdot b = \frac{rf(c_1, x, c_2)}{rx}(0) \cdot \left(\frac{rg(c_1, y, c_2)}{ry}(0) \cdot b \right) \quad (10)$$

PROOF: The proof is essentially the same as the equational calculations done in the proof of [2, Thm 37]. However, since it is crucial to proving the reverse Faà di Bruno's Formula, revisiting the proof using the term calculus may be useful. So first recall that since f is D-linear in A , it is also k -linear in A . So in particular $f(c_1, 0, c_2) = 0$. So then we compute:

$$\begin{aligned} \frac{rg(c_1, f(c_1, x, c_2), c_2)}{rx}(0) \cdot b &\stackrel{[\mathbf{R.5}]}{=} \frac{rf(c_1, x, c_2)}{rx}(0) \cdot \left(\frac{rg(c_1, y, c_2)}{ry}(f(c_1, 0, c_2)) \cdot b \right) \\ &\stackrel{\text{D-lin.} \Rightarrow k\text{-lin.}}{=} \frac{rf(c_1, x, c_2)}{rx}(0) \cdot \left(\frac{rg(c_1, y, c_2)}{ry}(0) \cdot b \right) \end{aligned}$$

So the desired equality holds. $\square$

Now [2, Thm 42] tells us that the D-linear transpose of the forward derivative is the reverse derivative, that is, $D[f]^\dagger[A \times \_] = R[f]$, which in term calculus is written as:

$$\frac{r \frac{df(x)}{dx}(a) \cdot u}{ru}(0) \cdot b = \frac{rf(x)}{rx}(a) \cdot b \quad (11)$$

which we note is precisely **[RD.6]**. Moreover, this also implies that $R[f]$ is D-linear in its second argument and that the D-linear transpose of the reverse derivative is the forward derivative, that is, $R[f]^\dagger[A \times \_] = D[f]$. Furthermore, we can also show that the partial reverse derivative in the second argument of the reverse derivative is precisely the forward derivative:

Lemma 3.4 *In a Cartesian k -reverse differential category, the following equalities hold:*

$$\frac{r \frac{df(x)}{dx}(a_1) \cdot u}{ru}(b) \cdot a_2 = \frac{df(x)}{dx}(a_1) \cdot a_2 \quad (12)$$

PROOF: We compute:

$$\frac{r \frac{rf(x)}{rx}(a_1) \cdot u}{ru}(b) \cdot a_2 \stackrel{(11)}{=} \frac{d \frac{rf(x)}{rx}(a_1) \cdot u}{du}(b) \cdot v \stackrel{(0)}{=} \frac{r \frac{rf(x)}{rx}(a_1) \cdot v}{rv}(0) \cdot a_2 \stackrel{\text{D-lin.}}{=} \frac{r \frac{rf(x)}{rx}(a_1) \cdot v}{rv}(0) \cdot a_2 \stackrel{(7)}{=} \frac{df(x)}{dx}(a_1) \cdot a_2$$

So the desired equality holds. $\square$

We conclude this section by showing that the partial reverse derivative is the D-linear transpose of the partial forward derivative.

Lemma 3.5 *In a Cartesian k -reverse differential category, for every map $f : A_1 \times \dots \times A_n \rightarrow B$, we have that $D_j[f]^\dagger[A_1 \times \dots \times A_n \times \_] = R_j[f]$, which in the term calculus is expressed as follows:*

$$\frac{r \frac{df(a_1, \dots, a_{j-1}, x_j, a_{j+1}, \dots, a_n)}{dx_j}(a_j) \cdot u}{ru}(0) \cdot b = \frac{rf(a_1, \dots, a_{j-1}, x_j, a_{j+1}, \dots, a_n)(a_j) \cdot b}{rx_j} \quad (13)$$

PROOF: Without loss of generality, for simplicity, we prove this for a map of type $f : C_1 \times A \times C_2 \rightarrow B$. So we compute that:

$$\begin{aligned} & \frac{r \frac{df(c_1, x, c_2)}{dx}(a) \cdot u}{ru}(0) \cdot b \stackrel{\text{Def.}}{=} \frac{r \frac{df(v, x, w)}{d(v, x, w)}(c_1, a, c_2) \cdot (0, u, 0)}{ru}(0) \cdot b \\ & \stackrel{(7)}{=} \frac{r \frac{rf(v, x, w)}{r(v, x, w)}(c_1, a, c_2) \cdot y}{ry}(0) \cdot (0, u, 0) \stackrel{\text{Def.}}{=} \frac{r \frac{rf(v, x, w)}{r(v, x, w)}(c_1, a, c_2) \cdot y}{ry}(0) \cdot (s, u, t) \stackrel{\pi_2}{=} \frac{r \frac{rf(v, x, w)}{r(v, x, w)}(c_1, a, c_2) \cdot y}{r(s, u, t)}(0, 0, 0) \cdot b \\ & \stackrel{[R.6]}{=} \pi_2 \left(\frac{rf(v, x, w)}{r(v, x, w)}(c_1, a, c_2) \cdot b \right) \stackrel{\text{Def.}}{=} \frac{rf(c_1, x, c_2)}{rx}(a) \cdot b \end{aligned}$$

So the desired equality holds. $\square$

4 Reverse Faà di Bruno's Formula

To provide a reverse differentiation version of Faà di Bruno's Formula, we must first work out the appropriate notion of *higher-order reverse derivatives*. Now first observe that given a map $f : A \rightarrow B$, reverse deriving it once gives $R[f] : A \times B \rightarrow A$, reverse deriving it again gives $R^2[f] : A \times B \times A \rightarrow A \times B$, then reverse deriving again gives $R^3[f] : A \times B \times A \times A \times B \rightarrow A \times B \times A$, so on and so forth. We quickly see that both the domain and codomain of $R^n[f]$ expand quite rapidly, and thus $R^n[f]$ is not necessarily easy to work with. However, it turns out that from Lemma 2.5 and Lemma 3.4, there is a lot of redundant information in $R^n[f]$. For example, we can compute that $R^2[f]$ is worked out to be:

$$\frac{r \frac{rf(x)}{rx}(y) \cdot z}{r(y, z)}(a_1, b) \cdot a_2 = \left\langle \frac{r \frac{rf(x)}{rx}(a_1) \cdot u}{ru}(b) \cdot a_2, \frac{df(x)}{dx}(a_1) \cdot a_2 \right\rangle$$

So we see that $R^2[f]$ contains previous known information, since $D[f]$ is the D-linear transpose of $R[f]$. So, all the new information comes from taking the partial reverse derivative in the first argument. This is how we get the higher-order reverse derivatives. So starting with a map $f : A \rightarrow B$, we again first take its reverse derivative $R[f] : A \times B \rightarrow A$, and then this time only take the partial reverse derivative in the first argument to get $R_1[R[f]] : A \times B \times A \rightarrow A$, then do this again to get $R_1[R_1[R[f]]] : A \times B \times A \times A \rightarrow A$, and so on. So after doing this $n + 1$ times, we get a map of type $A \times B \times A^{\times n} \rightarrow A$.

Definition 4.1 *In a Cartesian k -reverse differential category, for a map $f : A \rightarrow B$, the $n + 1$ -th reverse derivative of f is the map $\rho^{(n+1)}[f] : A \times B \times A^{\times n} \rightarrow A$ defined inductively as $\rho^{(1)}[f] = R[f]$ and $\rho^{(n+2)}[f] = R_1[\rho^{(n+1)}[f]]$.*

In the term calculus, inspired by the notation used in [4, Sec 3.1] for higher-order forward derivatives, we write higher-order reverse derivatives as follows:

$$\rho^{(n+1)}[f](a_0, b, a_2, \dots, a_{n+1}) = \frac{r^{(n+1)}f(x)}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1}$$

As such, by definition, we have that:

$$\begin{aligned} \frac{r^{(1)}f(x)}{rx}(a_0) \cdot b &= \frac{rf(x)}{rx}(a_0) \cdot b \\ \frac{r^{(n+2)}f(x)}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} &= \frac{\frac{r^{(n+1)}f(x)}{rx}(y) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1}}{ry}(a_0) \cdot a_{n+2} \end{aligned}$$

Higher-order forward derivatives are defined similarly. Starting again with a map $f : A \rightarrow B$, we can repeatedly derive the first argument to get a map of type $A \times A^{\times n+1} \rightarrow B$. So the $n + 1$ -th **forward derivative** [11, Def 3.1] of f is the map $\partial^{(n+1)}[f] : A \times A^{\times n} \rightarrow B$, which is defined inductively as $\partial^{(1)}[f] = D[f]$ and $\partial^{(n+2)}[f] = D_1[\partial^{(n+1)}[f]]$. In the term calculus, following the notation used in [4, Sec 3.1], we write higher-order forward derivatives as:

$$\partial^{(n+1)}[f](a_0, a_1, \dots, a_{n+1}) = \frac{d^{(n+1)}f(x)}{dx}(a_0) \cdot a_1 \cdot \dots \cdot a_{n+1}$$

So, by definition, we have that:

$$\begin{aligned} \frac{d^{(1)}f(x)}{dx}(a_0) \cdot b &= \frac{df(x)}{dx}(a_0) \cdot b \\ \frac{d^{(n+2)}f(x)}{dx}(a_0) \cdot a_1 \cdot a_2 \cdot \dots \cdot a_{n+1} &= \frac{\frac{d^{(n+1)}f(x)}{dx}(y) \cdot a_1 \cdot a_2 \cdot \dots \cdot a_{n+1}}{dy}(a_0) \cdot a_{n+2} \end{aligned}$$

Now the $n + 1$ -th forward derivative is D-linear in each of the last $n + 1$ arguments A and is also symmetric in its last $n + 1$ arguments [11, Lemma 3.1.(i)]. As such, it does not matter which D-linear argument we transpose. So for convenience, to line up with the type higher-order reverse derivative, we will consider the D-linear transpose of the second argument of the higher-order forward derivative: $\partial^{(n+1)}[f]^{\dagger[A \times \_ \times A^n]} : A \times B \times A^{\times n} \rightarrow A$. Now we'd like that $\partial^{(n+1)}[f]^{\dagger[A \times \_ \times A^n]}$ be equal to $\rho^{(n+1)}$. Unfortunately, this does not seem to necessarily follow from just [RD.1]-[RD.7]. As such, we introduce an extra compatibility between the forward differential combinator and the reverse differential combinator.

Given a map $f : A \rightarrow B$, we can take its forward derivative $D[f] : A \times A \rightarrow B$, and then take its reverse derivative in the first argument to get $R_1[D[f]] : A \times A \times B \rightarrow A$. On the other hand, we could have first taken the reverse derivative $R[f] : A \times B \rightarrow A$, and then taken its reverse derivative in the first argument to get $R_1[R[f]] : A \times B \times A \rightarrow A$. Up to swapping the last two arguments, we ask that these be equal.

Definition 4.2 A Cartesian k -reverse differential category is said to satisfy the **stable rule** if:

$$\frac{r \frac{df(x)}{dx}(y) \cdot a_2}{ry}(a_1) \cdot b = \frac{r \frac{f(x)}{rx}(y) \cdot b}{ry}(a_1) \cdot a_2 \quad (14)$$

To the best of our understanding at this moment, the stable rule does not seem to follow from the reverse differential combinator axioms – though we do not have a proper counter-example at present. The stable rule is a desirable coherence that should hold, and all natural examples of Cartesian reverse differential categories satisfy the stable rule. In future work, it may be worth revisiting the axioms of the reverse differential combinator to include the stable rule, which we conjecture is related to taking multiple linear transposes of multi-D-linear maps.

Example 4.3 Let us show that smooth functions satisfy the stable rule. For a smooth function $F = \langle f_1, \dots, f_m \rangle : \mathbb{R}^n \rightarrow \mathbb{R}^m$, recall that $R[F] : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $D[F] : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ are:

$$R[F](\vec{x}, \vec{y}) = (R_1[F](\vec{x}, \vec{y}), \dots, R_n[F](\vec{x}, \vec{y})) = \left(\sum_{j=1}^m \frac{\partial f_j}{\partial x_1}(\vec{x}) y_j, \dots, \sum_{j=1}^m \frac{\partial f_j}{\partial x_n}(\vec{x}) y_j \right)$$

$$D[F](\vec{x}, \vec{z}) = (D[f_1](\vec{x}, \vec{z}), \dots, D[f_m](\vec{x}, \vec{z})) = \left(\sum_{i=1}^n \frac{\partial f_1}{\partial x_i}(\vec{x}) z_i, \dots, \sum_{i=1}^n \frac{\partial f_m}{\partial x_i}(\vec{x}) z_i \right)$$

On the other hand, for a smooth function $G = \langle g_1, \dots, g_k \rangle : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^k$, its partial reverse derivative in its first argument $\mathbb{R}^n$ is $R_1[G] : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \rightarrow \mathbb{R}^n$ given as follows:

$$R_1[G](\vec{x}, \vec{y}, \vec{z}) = \left(\sum_{j=1}^k \frac{\partial g_j}{\partial x_1}(\vec{x}, \vec{y}) z_j, \dots, \sum_{j=1}^k \frac{\partial g_j}{\partial x_n}(\vec{x}, \vec{y}) z_j \right)$$

We need to show that $R_1[D[F]] : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ and $R_1[R[F]] : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ are equal up to permutation of the second and third arguments. We first compute $R_1[D[F]]$ to be:

$$\begin{aligned} R_1[D[F]](\vec{x}, \vec{z}, \vec{y}) &= \left(\sum_{j=1}^m \frac{\partial D[f_j]}{\partial x_1}(\vec{x}, \vec{z}) y_j, \dots, \sum_{j=1}^m \frac{\partial D[f_j]}{\partial x_n}(\vec{x}, \vec{z}) y_j \right) \\ &= \left(\sum_{j=1}^m \sum_{i=1}^n \frac{\partial^2 f_j}{\partial x_i \partial x_1}(\vec{x}) z_i y_j, \dots, \sum_{j=1}^m \sum_{i=1}^n \frac{\partial^2 f_j}{\partial x_i \partial x_n}(\vec{x}) z_i y_j \right) \end{aligned}$$

On the other hand, we compute $R_1[R[F]]$ to be:

$$\begin{aligned} R_1[R[F]](\vec{x}, \vec{y}, \vec{z}) &= \left(\sum_{i=1}^n \frac{\partial R_i[F]}{\partial x_1}(\vec{x}) z_i, \dots, \sum_{i=1}^n \frac{\partial R_i[F]}{\partial x_n}(\vec{x}) z_i \right) \\ &= \left(\sum_{i=1}^n \sum_{j=1}^m \frac{\partial^2 f_j}{\partial x_i \partial x_1}(\vec{x}) y_j z_i, \dots, \sum_{i=1}^n \sum_{j=1}^m \frac{\partial^2 f_j}{\partial x_i \partial x_n}(\vec{x}) y_j z_i \right) \end{aligned}$$

So we have that $R_1[D[F]](\vec{x}, \vec{z}, \vec{y}) = R_1[R[F]](\vec{x}, \vec{y}, \vec{z})$, and thus the stable rule holds in SMOOTH. We note that the same proof works for polynomials over any commutative semiring.

With the stable rule, we can show that D-linear transpose of the higher-order forward derivative is the higher-order reverse derivative. For convenience, by convention we define $\rho^{(0)}[f] = f = \partial^{(0)}[f]$.

Lemma 4.4 *In a Cartesian k -reverse differential category whose reverse differential combinator satisfies the stable rule, we have that $\partial^{(n+1)}[f]^{\dagger[A \times \_ \times A^n]} = \rho^{(n+1)}[f]$, which in the term calculus is expressed as:*

$$\frac{r \frac{d^{(n+1)}f(x)}{dx}(a_0) \cdot y \cdot a_2 \cdot \dots \cdot a_{n+1}}{ry}(0) \cdot b = \frac{r^{(n+1)}f(x)}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} \quad (15)$$

PROOF: One can show that the stable rule also holds in context, that is:

$$\frac{r \frac{dg(c_1, x, c_2)}{x}(y) \cdot a_2}{ry}(a_1) \cdot b = \frac{r \frac{rg(c_1, x, c_2)}{x}(y) \cdot b}{dy}(a_1) \cdot a_2 \quad (16)$$

Now note that the stable rule can be rewritten as:

$$\frac{r \frac{df(x)}{x}(y) \cdot a_2}{ry}(a_1) \cdot b = \frac{r^{(2)}f(x)}{d(x)}(a_1) \cdot b \cdot a_2 \quad (17)$$

Since the stable rule holds in context, it is straightforward to show by induction that:

$$\frac{r \frac{d^{(n)}f(x)}{x}(y) \cdot a_1 \cdot \dots \cdot a_n}{ry}(a_0) \cdot b = \frac{r^{(n+1)}f(x)}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} \quad (18)$$

Then applying (11) to the left-hand side gives us precisely that:

$$\begin{aligned} & \frac{r \frac{d^{(n+1)}f(x)}{dx}(a_0) \cdot y \cdot a_2 \cdot \dots \cdot a_{n+1}}{ry}(0) \cdot b \stackrel{\text{Def.}}{=} \frac{r \frac{d \frac{d^{(n)}f(x)}{x}(y) \cdot a_1 \cdot \dots \cdot a_n}{dy}(a_0) \cdot y}{ry}(0) \cdot b \\ & \stackrel{(11)}{=} \frac{r \frac{d^{(n)}f(x)}{x}(y) \cdot a_1 \cdot \dots \cdot a_n}{ry}(a_0) \cdot b \stackrel{(18)}{=} \frac{r^{(n+1)}f(x)}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} \end{aligned}$$

So we get that $\partial^{(n+1)}[f]^{\dagger[A \times \_ \times A^n]} = \rho^{(n+1)}[f]$ as desired. $\square$

We may finally look towards understanding the reverse Faà di Bruno's Formula. Let us first review Faà di Bruno's Formula for forward differentiation in a Cartesian differential category. To do so, we first introduce some notation. For every $n \in \mathbb{N}$, define the well-ordered set $[n+1] = \{1 < \dots < n+1\}$. Now for every subset $I = \{i_1 < \dots < i_m\} \subseteq [n+1]$, for a vector $\vec{x} = (x_1, \dots, x_{n+1})$, define $x_I = (x_{i_1}, \dots, x_{i_m})$. Lastly, we denote a *non-empty* partition of $[n+1]$ as $[n+1] = S_1 \mid \dots \mid S_k$, and let $|S_j|$ be the cardinality of S_j . Then Faà di Bruno's Formula [11, Lemma 3.4] for the $n+1$ -th derivative is given as a sum over the non-empty partitions of $[n+1]$ as follows:

$$\begin{aligned} & \frac{d^{(n+1)}g(f(x))}{dx}(a_0) \cdot a_1 \cdot a_2 \cdot \dots \cdot a_{n+1} \\ &= \sum_{[n+1]=S_1 \mid \dots \mid S_k} \frac{d^{(k)}g(z)}{dz}(f(a_0)) \cdot \left(\frac{d^{(|S_1|)}f(x)}{dx}(a_0) \cdot a_{S_1} \right) \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right) \end{aligned} \quad (19)$$

We are finally in a position to work out the reverse Faà di Bruno's Formula. For convenience, we will assume, without loss of generality, that for each non-empty partition $[n+1] = S_1 | \dots | S_k$, that $1 \in [n+1]$ is always in $1 \in S_1$. As a shorthand, we write $\widehat{S}_1 = S_1 - \{1\}$.

Theorem 4.5 *In a Cartesian k -reverse differential category whose reverse differential combinator satisfies the stable rule, the following equality holds:*

$$\begin{aligned} & \frac{r^{(n+1)}g(f(x))}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} = \\ & \sum_{\substack{[n+1]=S_1|\dots|S_k \\ 1 \in A_1}} \frac{r^{(|S_1|)}f(x)}{rx}(a_0) \cdot \left(\frac{r^{(k)}g(y)}{ry}(a_0) \cdot b \cdot \left(\frac{d^{(|S_2|)}f(x)}{dx}(a_0) \cdot a_{S_2} \right) \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right) \right) \cdot a_{\widehat{S}_1} \end{aligned} \quad (20)$$

PROOF: We compute:

$$\begin{aligned} & \frac{r^{(n+1)}g(f(x))}{rx}(a_0) \cdot b \cdot a_2 \cdot \dots \cdot a_{n+1} \stackrel{(15)}{=} \frac{r^{(n+1)}g(f(x))}{dx}(a_0) \cdot y \cdot a_2 \cdot \dots \cdot a_{n+1} \cdot b \\ & \stackrel{(19)}{=} \frac{r \left(\sum_{[n+1]=S_1|\dots|S_k} \frac{d^{(k)}g(z)}{dz}(f(a_0)) \cdot \left(\frac{d^{(|S_1|)}f(x)}{dx}(a_0) \cdot y \cdot a_{S_1-\{1\}} \right) \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right) \right)}{ry} \cdot (0) \cdot b \\ & \stackrel{[R.2]}{=} \sum_{[n+1]=S_1|\dots|S_k} \frac{r \frac{d^{(k)}g(z)}{dz}(f(a_0)) \cdot \left(\frac{d^{(|S_1|)}f(x)}{dx}(a_0) \cdot y \cdot a_{S_1-\{1\}} \right) \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right)}{ry} \cdot (0) \cdot b \\ & \stackrel{(10)}{=} \sum_{\substack{[n+1]=S_1|\dots|S_k \\ 1 \in A_1}} \frac{r \frac{d^{(|S_1|)}f(x)}{dx}(a_0) \cdot y \cdot a_{S_1-\{1\}}}{ry} \cdot (0) \cdot \left(\frac{r \frac{d^{(k)}g(z)}{dz}(f(a_0)) \cdot z \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right)}{rz} \cdot (0) \cdot b \right) \\ & \stackrel{(15)}{=} \sum_{\substack{[n+1]=S_1|\dots|S_k \\ 1 \in A_1}} \frac{r^{(|S_1|)}f(x)}{rx}(a_0) \cdot \left(\frac{r^{(k)}g(y)}{ry}(a_0) \cdot b \cdot \left(\frac{d^{(|S_2|)}f(x)}{dx}(a_0) \cdot a_{S_2} \right) \cdot \dots \cdot \left(\frac{d^{(|S_k|)}f(x)}{dx}(a_0) \cdot a_{S_k} \right) \right) \cdot a_{\widehat{S}_1} \end{aligned}$$

So, the reverse of Faà di Bruno's Formula holds as desired. $\square$

When $n = 0$ in the reverse Faà di Bruno's Formula, we get back precisely the reverse chain rule. Indeed, the only non-empty partition of $[1]$ is $[1] = S_1 = \{1\}$, which is why no forward derivatives appear in the reverse chain rule. When $n = 1$, the non-empty partitions of $[2]$ are $[1] = S_1 = \{1\} | S_2 = \{2\}$ and $[2] = S_1 = \{1, 2\}$. So the reverse Faà di Bruno's Formula for the second reverse derivative is:

$$\begin{aligned} & \frac{r^{(2)}g(f(x))}{rx}(a_0) \cdot b \cdot a_2 \\ & = \frac{r^{(1)}f(x)}{rx}(a_0) \cdot \left(\frac{r^{(2)}g(y)}{ry}(f(a_0)) \cdot b \cdot \left(\frac{df(x)}{dx}(a_0) \cdot a_2 \right) \right) + \frac{r^{(2)}f(x)}{rx}(a_0) \cdot \left(\frac{r^{(1)}g(y)}{ry}(f(a_0)) \cdot b \right) \cdot a_2 \end{aligned}$$

The reverse Faà di Bruno's Formula for the third reverse derivative will involve five summands, the formula for the fourth reverse derivative will involve fifteen summands, etc.

References

- [1] R. F. Blute, J. R. B. Cockett & R. A. G. Seely (2009): *Cartesian Differential Categories*. *Theory and Applications of Categories* 22(23), pp. 622–672. Available at <http://www.tac.mta.ca/tac/volumes/22/23/22-23abs.html>.
- [2] J. R. B. Cockett, G. S. H. Cruttwell, J. D. Gallagher, J.-S. P. Lemay, B. Mac Adam, G. Plotkin & D. Pronk (2020): *Reverse Derivative Categories*. *LIPIcs* 152(CSL 2020), pp. 18:1–18:16, doi:10.4230/LIPIcs.CSL.2020.18.
- [3] J. R. B. Cockett & J.-S. P. Lemay (2022): *Linearizing Combinators*. *Theory and Applications of Categories* 38(13), pp. 374–431. Available at <http://www.tac.mta.ca/tac/volumes/38/13/38-13abs.html>.
- [4] J. R. B. Cockett & R. A. G. Seely (2011): *The Faà di Bruno construction*. *Theory and Applications of Categories* 25(15), pp. 394–425, Available at <http://www.tac.mta.ca/tac/volumes/25/15/25-15abs.html>.
- [5] G. Cruttwell, J. Gallagher, J.-S. P. Lemay & D. Pronk (2022): *Monoidal reverse differential categories*. *Mathematical Structures in Computer Science* 32(10), pp. 1313–1363, doi:10.1017/S096012952200038X.
- [6] G. Cruttwell, J. Gallagher & B. MacAdam (2019): *Towards formalizing and extending differential programming using tangent categories*. *Proc. ACT 2019*. Available at <https://www.cs.ox.ac.uk/ACT2019/preproceedings/Jonathan%20Gallagher,%20Geoff%20Cruttwell%20and%20Ben%20MacAdam.pdf>.
- [7] G. Cruttwell, J. Gallagher & D. Pronk (2021): *Categorical Semantics of a Simple Differential Programming Language*. *Electronic Proceedings in Theoretical Computer Science* 333, pp. 289–310, doi:10.4204/EPTCS.333.20.
- [8] G. Cruttwell & J.-S. P. Lemay (2024): *Reverse Tangent Categories*. In Aniello Murano & Alexandra Silva, editors: *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 288, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 21:1–21:21, doi:10.4230/LIPIcs.CSL.2024.21.
- [9] G.S.H. Cruttwell, B. Gavranovic, N. Ghani, P. Wilson & F. Zanasi (2022): *Categorical Foundations of Gradient-Based Learning*. *ESOP 2022*, pp. 1–28, doi:10.1007/978-3-030-99336-8_1.
- [10] B. Fong, D. Spivak & R. Tuyéras (2019): *Backprop as functor: A compositional perspective on supervised learning*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, pp. 1–13, doi:10.5555/3470152.3470163.
- [11] R. Garner & J.-S. P. Lemay (2021): *Cartesian differential categories as skew enriched categories*. *Applied Categorical Structures* 29(6), pp. 1099–1150, doi:10.1007/s10485-021-09649-7.
- [12] B. Gavranović, P. Lessard, A. Dudzik, T. von Glehn, J. Araújo & P. Veličković (2024): *Categorical Deep Learning: An Algebraic Theory of Architectures*. *arXiv preprint arXiv:2402.15332*, doi:10.48550/arXiv.2402.15332.
- [13] J.-S. P. Lemay (2022): *Jacobians and Gradients for Cartesian Differential Categories*. In Kohei Kishida, editor: *Proceedings of the Fourth International Conference on Applied Category Theory*, Cambridge, United Kingdom, 12-16th July 2021, *Electronic Proceedings in Theoretical Computer Science* 372, Open Publishing Association, pp. 29–42, doi:10.4204/EPTCS.372.3.
- [14] M. Vákár & T. Smeding (2022): *CHAD: Combinatory homomorphic automatic differentiation*. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 44(3), pp. 1–49, doi:10.1145/3527634.
- [15] P. Wilson & F. Zanasi (2021): *Reverse Derivative Ascent: A Categorical Approach to Learning Boolean Circuits*. In: *Proceedings of the 3rd Annual International Applied Category Theory Conference 2020*, Cambridge, USA, 6-10th July 2020, 333, Open Publishing Association, pp. 247–260, doi:10.4204/EPTCS.333.17.
- [16] P. Wilson & F. Zanasi (2022): *Categories of differentiable polynomial circuits for machine learning*. In: *International Conference on Graph Transformation*, Springer, pp. 77–93, doi:10.1007/978-3-031-09843-7_5.

Higher-Order DisCoCat

(*Peirce-Lambek-Montague semantics*)

Alexis Toumi Giovanni de Felice

Quantinuum – Quantum Compositional Intelligence
17 Beaumont street, OX1 2NA Oxford, UK

We propose a new definition of higher-order DisCoCat (categorical compositional distributional) models where the meaning of a word is not a diagram, but a diagram-valued higher-order function. Our models can be seen as a variant of Montague semantics based on a lambda calculus where the primitives act on string diagrams rather than logical formulae. As a special case, we show how to translate from the Lambek calculus into Peirce’s system beta for first-order logic. This allows us to give a purely diagrammatic treatment of higher-order and non-linear processes in natural language semantics: adverbs, prepositions, negation and quantifiers. The definition presented in this article comes with a proof-of-concept implementation in DisCoPy, the Python library for string diagrams.

Introduction

DisCoCat [6, 7] (Categorical Compositional Distributional) models are structure-preserving maps which send *grammatical types to vector spaces* and *grammatical structures to linear maps*. Concretely, the meaning of words is given by tensors with shapes induced by their grammatical types; the meaning of sentences is given by contracting the tensor networks induced by their grammatical structure. *String diagrams* provide an intuitive graphical language to visualise and reason formally about the evaluation of DisCoCat models; which can be formalised in terms of *functors* $F : \mathbf{G} \rightarrow \mathbf{Vect}$ from the category generated by a formal grammar $\mathbf{G}$ to the monoidal category $\mathbf{Vect}$ of vector spaces and linear maps with the tensor product [11, §2.5].

Although this functorial definition applies equally to any kind of formal grammar, most of the DisCoCat literature focuses on *pregroup grammars* and more generally on *categorical grammars* such as the *Lambek calculus* [18, 10] and *combinatory categorical grammars* (CCG) [28]. In that case, $\mathbf{G}$ is a *closed monoidal category* and the DisCoCat models $F : \mathbf{G} \rightarrow \mathbf{Vect}$ map grammatical structures to the closed structure of $\mathbf{Vect}$ in a canonical way. In practice, this means that once the meaning of each word is computed from a dataset, the meaning of any new grammatical sentence can be computed automatically from its grammatical structure.

While assigning vectors to words of *atomic types* like common nouns was already a standard technique in the field of distributional semantics, the main novelty of the DisCoCat framework lies in its treatment of *complex types* like adjectives, verbs, etc. If the atomic type n for common nouns is assigned a vector space $F(n) = N$, the complex type $n \leftarrow n$ for adjectives (modifying nouns on their right) is mapped to the tensor product $F(n \leftarrow n) = N \otimes N^*$: *adjectives are not vectors, they are linear maps*. Now if we play the same game with the type for adverbs (modifying adjectives on their right) we get tensors of order four: $F((n \leftarrow n) \leftarrow (n \leftarrow n)) = N \otimes N^* \otimes N^* \otimes N$. Not only does this lead to unreasonably large representations — if we want 1000 parameters for each noun, we need 1000 billion parameters for *each adverb* — but we argue that it also misses the point of such higher-order types altogether.

Because the codomain of DisCoCat models $F : \mathbf{G} \rightarrow \mathbf{Vect}$ is a *compact-closed category*, any second-order type $a \leftarrow (b \leftarrow c)$ is squashed down to the same vector space $A \otimes (B \otimes C^*)^* \simeq A \otimes C \otimes B^*$ as the

first-order types $a(c \leftarrow b)$ and $(ac) \leftarrow b$. This collapse down to first-order makes it impossible to express linguistic phenomena that are inherently higher-order¹ such as negation, disjunction and universal quantification. Indeed, the variant of DisCoCat models valued in the compact closed category of relations **Rel** are as expressive as *regular logic*, the fragment of first-order logic generated by truth, conjunction and existential quantification [14]. The one missing ingredient to get from regular logic to all of first-order logic is *negation*, which cannot be expressed in terms of relational composition. In short, *negation cannot be a relation* but it can be expressed as a non-linear, higher-order function which maps relations to relations.

This observation can be traced back to Peirce [21]’s *system beta* and his *existential graphs* which are both the first examples of string diagrams and arguably the first axiomatisation of first-order logic. In today’s terminology, Peirce defined string diagrams with *swaps*, *cups*, *caps*, *spiders* — which Peirce called *lines of identity* — and a *unary operator on homsets* for negation represented as a circle that he called *cut*. Existential graphs come with graphical equations that are sound and complete for first-order logic, see [4, 15] for a categorical presentation.

Higher-order functions are at the basis of *Montague semantics* [20]. Concretely, the meaning of a word is given by a term in the simply-typed lambda calculus with the symbols of first-order logic² taken as primitives; the meaning of a sentence is given by composing these lambda terms together to get a closed formula. Abstractly, this can be formalised as a closed monoidal functor $F : \mathbf{G} \rightarrow \Lambda L$ where $\mathbf{G}$ is a categorial grammar and L is the language of first-order logic (predicates, negation, conjunction, quantifiers, etc.) seen as the presentation of a cartesian closed category ΛL [11, §2.2]. Interpretations of the logic in a model then induce closed monoidal functors $[[-]] : \Lambda L \rightarrow \mathbf{Set}$, which we can compose with F to get a set-theoretic interpretation of the grammar $[[-]] \circ F : \mathbf{G} \rightarrow \mathbf{Set}$.

Let us summarise the situation. On the one hand, we have DisCoCat models defined in terms of tensors that can be learnt from data and compared efficiently (with e.g. inner products) but that cannot express higher-order phenomena such as negation and universal quantification. On the other, we have the higher-order functions of Montague semantics, but where the resulting logical formulae are not easily amenable to machine learning and computationally intractable (e.g. equivalence of first-order logic formulae is undecidable). What we propose is to get the best of both worlds. As in Montague semantics, we assign a simply-typed lambda term to each word, but now with diagram-valued operations as primitives. The meaning of a sentence is given by composing these lambda terms together to get a diagram which can be interpreted in **Vect**, as in a DisCoCat model, or in fact in any monoidal category $\mathbf{C}$.

We propose to go from standard DisCoCat models $F : \mathbf{G} \rightarrow \mathbf{Vect}$ to *higher-order DisCoCat* (HO-DisCoCat) models defined as closed monoidal functors $F : \mathbf{G} \rightarrow \Lambda D$ where D is the language of *string diagrams* (boxes, identity, composition, tensor, etc.) again seen as the signature for a free cartesian closed category ΛD . The closed monoidal functors $[[-]] : \Lambda D \rightarrow \mathbf{Set}$ now correspond precisely to interpretations of the diagrams in any monoidal category $\mathbf{C}$. Thus we can replace **Vect** by any monoidal category $\mathbf{C}$, which need not be compact closed. This allows many new variants of DisCoCat models, for example based on neural networks or stochastic processes. It also has potential applications in quantum natural language processing [9] (QNLP) where the compact closed structure comes at the cost of exponentially low probability of success [25].

The rest of this article is organised as follows. Section 1 gives a first toy example of a higher-order

¹Here we mean higher-order in the sense of functions taking functions as input, although we are still within the realm of first-order logic with quantifiers over variables rather than predicates. There is yet another ambiguous meaning of order: an element of $A \otimes C \otimes B^*$ is called a tensor of order three but it is still a first-order function when seen as a linear map $A \otimes C \rightarrow B$.

²Montague semantics dealt with first-order *modal logic*, we leave the treatment of modalities in natural language using Peirce’s system gamma as a promising direction for future work.

process which cannot be accounted for in these standard DisCoCat models. Section 2 formulates a standard example of Montague semantics in terms of a closed monoidal functor $F : \mathbf{G} \rightarrow \Lambda L$. Section 3 contains our main definition: HO-DisCoCat models as closed monoidal functors $F : \mathbf{G} \rightarrow \Lambda D$. Section 4 shows how our models can be extended to monoidal categories with additional operations, which we use to model arbitrary adverbs depicted as *boxes with holes* that can hold diagrams themselves. Section 5 defines a HO-DisCoCat model which we call *Peirce-Lambek-Montague* semantics, where the diagrams are existential graphs. This closes the conjecture opened in [6]: DisCoCat models valued in **Rel** are not as expressive as Montague semantics, but higher-order DisCoCat models are. Appendix A showcases an implementation of this Peirce-Lambek-Montague semantics using DisCoPy [12], the Python toolkit for computing with string diagrams.

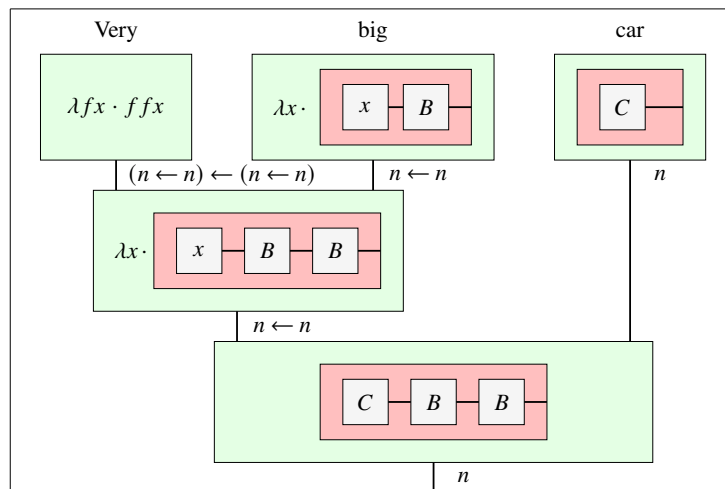
1 A toy example: “very big” = “big big”

To keep things concrete, we delay the formal definition of HO-DisCoCat models and start with a simple example of a higher-order function: the function $\text{twice} : (X \rightarrow X) \rightarrow (X \rightarrow X)$ which takes a function and applies it twice, i.e. $\text{twice} = \lambda f x. f f x$. In our toy model of semantics:

- the noun “car” is sent to some vector $F(\text{car}) \in N$ drawn as a box with one wire out,
- the adjective “big” is sent to the function $F(\text{big}) = \lambda x. B \circ x$ which post-composes a vector x with some linear map B , drawn as a box with one wire in and one wire out,
- the adverb “very” is sent to the non-linear, higher-order function $F(\text{very}) = \text{twice}$.

We can now apply it to the noun phrase “very big car” and check that indeed it has the same semantics as “big big car”. The image of the functor $F : \mathbf{G} \rightarrow \Lambda D$ on a syntax tree can be visualised by annotating each node with a lambda term with string diagrams as primitives. The leaves of the tree contain the lambda terms for each word. Each binary node is the application of a subtree of type $x \leftarrow y$ on the left ($y \rightarrow x$ on the right) to one of type y on the other side.

Remark. The following notation for syntax trees annotated with their semantics is borrowed from the one used for CCG derivations [3], except that we use string diagrams rather than the traditional predicate-argument structure. These are not quite string diagrams in the category **Set**: boxes are not labeled with their corresponding function, but with the result of applying this function to its arguments.



Due to the categorical no-cloning theorem [1], copying an adjective in this way is not possible in a standard DisCoCat model $F : \mathbf{G} \rightarrow \mathbf{Vect}$. Indeed, if $\dim(X) \geq 2$ there is no linear map $\text{twice} : X \otimes X^* \rightarrow X \otimes X^*$ such that $\text{twice}(f) = f \circ f$ for all linear maps $f : X \rightarrow X$. How can our higher-order DisCoCat model $F : \mathbf{G} \rightarrow \Lambda D$ get away with such a fundamental law of physics? To clear any such confusion, it is important to distinguish the three categories at play in the example above:

1. The (white) closed monoidal category $\mathbf{G}$ generated by arrows $w \rightarrow t$ for each word w of type t . The arrows $w_1 \dots w_n \rightarrow s$ are the syntax trees of grammatical sentences.
2. The (green) cartesian closed category ΛD where the arrows are lambda terms with the composition of string diagrams as primitives. Thanks to the cartesian product, lambda terms can copy and discard variables at will.
3. The (pink) monoidal category $\mathbf{Vect}$ where the computation of the model happens. This can be replaced by any monoidal category $\mathbf{C}$, be it cartesian, compact or otherwise.

Remark. *The equivalence between the lambda calculus with simple types and cartesian closed categories is known as the Curry-Howard-Lambek isomorphism [2, §1.6]. The correspondence between categorial grammars and closed monoidal categories is due to Lambek [17].*

Thus, higher-order DisCoCat models allow to make a distinction between the resource-sensitive world of $\mathbf{Vect}$ where computation happens and the cartesian category ΛD where we can do arbitrary meta-computation. This may be summarised in a slogan: *processes cannot copy, but we can copy processes*. Take the example of quantum circuits: there is no circuit that can copy arbitrary qubits, but we can copy the classical instructions for any quantum circuit.

2 Montague semantics as a closed monoidal functor

Before we define HO-DisCoCat and its string-diagram-valued lambda calculus, we first look at the simpler case of standard Montague semantics, defined as a monoidal functor $F : \mathbf{G} \rightarrow \Lambda L$. We adapt the presentation from [26].

As codomain ΛL , we take the cartesian closed category with objects generated by $L_0 = \{\tau, \phi\}$ for *terms* and *formulae*. The generating arrows L_1 are listed in the table below, they are chosen so that the arrows $f : \tau^{\otimes k} \rightarrow \phi$ in ΛL are all the well-formed formulae of first-order logic with $k \geq 0$ free variables. The relations $(f, f') \in L_2$ between pairs of formulae $f, f' : \tau^{\otimes k} \rightarrow \phi$ may be given by any sound and complete deduction system for first-order logic, so that $f = f'$ in ΛL if and only if f and f' are equivalent formulae. The closed monoidal functors $[[-]] : \Lambda L \rightarrow \mathbf{Set}$ are *interpretations*, also called first-order *structures*: they are defined by a set $[[\tau]]$ called the *universe* and a set $[[\phi]]$ of *truth values*, together with an element $[[\text{Alice}]] \in [[\tau]]$ for each constant, a function $[[\text{mortal}]] : [[\tau]] \rightarrow [[\phi]]$ for each unary predicate, etc.

L_1	symbol	type
constants	Alice, Bob, ...	τ
unary predicates	man, sings, ...	$\tau \rightarrow \phi$
binary predicates	needs, sees, ...	$\tau \rightarrow (\tau \rightarrow \phi)$
nullary operators	$\top, \perp$	ϕ
unary operators	$\neg$	$\phi \rightarrow \phi$
binary operators	$\wedge, \vee, \rightarrow$	$\phi \rightarrow (\phi \rightarrow \phi)$
quantifiers	$\forall, \exists$	$(\tau \rightarrow \phi) \rightarrow \phi$

Remark. the quantifiers $\forall, \exists$ are higher-order functions $(\tau \rightarrow \phi) \rightarrow \phi$, they take a unary predicate $\tau \rightarrow \phi$ (i.e. a formula with a free variable) and return a formula in ϕ . Thus, a first-order logic formula is a closed lambda term with generators in L_1 where every λ is preceded by $\forall$ or $\exists$. The variables of first-order logic are encoded as variables of the lambda calculus, with implicit α -equivalence so that e.g. $(\exists x.Px) = (\exists y.Py)$.

Remark. The presentation L implicitly depends on a first-order signature which contains all the non-logical symbols (constants and predicates).

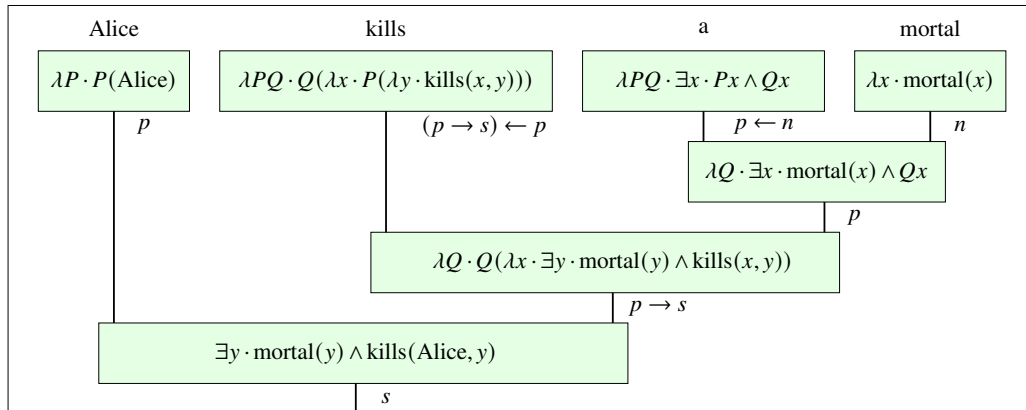
As domain $\mathbf{G}$, we take the free closed monoidal category with objects generated by the grammatical types $\{s, n, p\}$ for *sentence*, *common noun* and *noun phrase* respectively, together with the words in some finite vocabulary $w \in V$. There is a generating arrow $w \rightarrow t$ for each *dictionary entry*, i.e. for each word $w \in V$ of type t as listed in the table below. We define the image of our functor on generating objects as follows: sentences go to formulae, common nouns go to predicates and noun phrases go to functions from predicates to formulae. For example, the function for the noun phrase “no man” takes as input any formula $P(x)$ with a free variable and return the closed formula $\forall x. \neg \text{man}(x) \vee P(x)$.

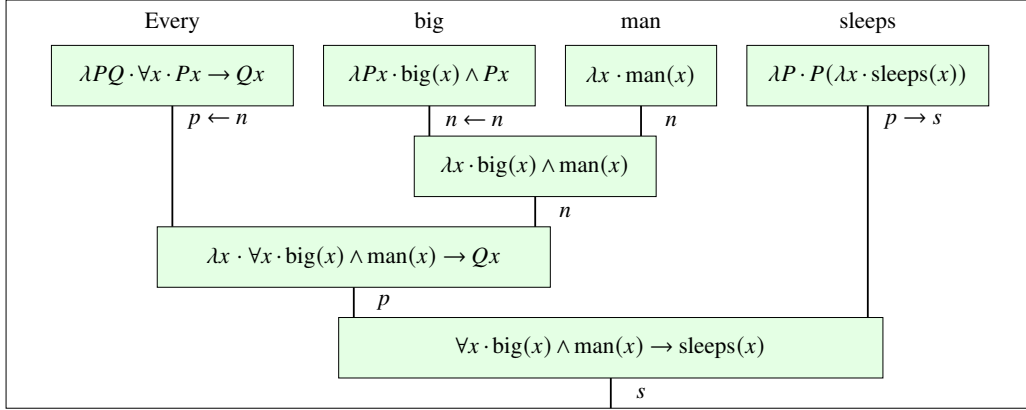
$$F(s) = \phi \quad F(n) = \tau \rightarrow \phi \quad F(p) = (\tau \rightarrow \phi) \rightarrow \phi$$

We assume that the objects for each word go to the monoidal unit, i.e. $F(w) = 1$ so that the meaning of dictionary entries $w \rightarrow t$ is given by the states $F(w \rightarrow t) : 1 \rightarrow F(t)$ listed below.

Montague semantics	word w	type t	meaning $F(w \rightarrow t)$
proper nouns	Alice	p	$\lambda P.P(\text{Alice})$
common nouns	man	n	$\lambda x.\text{man}(x)$
adjectives	big	$n \leftarrow n$	$\lambda Px.\text{big}(x) \wedge Px$
determiners	every	$p \leftarrow n$	$\lambda PQ.\forall x.Px \rightarrow Qx$
intransitive verbs	sleeps	$p \rightarrow s$	$\lambda P.P(\lambda x.\text{sleeps}(x))$
transitive verbs	kills	$(p \rightarrow s) \leftarrow p$	$\lambda PQ.Q(\lambda x.P(\lambda y.\text{kills}(x, y)))$

Again, we can visualise the action of the functor $F : \mathbf{G} \rightarrow \Lambda L$ on the syntax trees for a grammatical sentence by labeling each node with a lambda term. Note how the meanings of determiners “every” and “a” use both the higher-order and cartesian structure of the cartesian closed category ΛL : they take two predicates P and Q , quantify over a variable x of which they make two copies, feeding one to P and the other to Q before combining the resulting formulae.





3 Higher-order DisCoCat as a closed monoidal functor

We can now get to our main definition: HO-DisCoCat models as closed monoidal functors $F : \mathbf{G} \rightarrow \Lambda D$ from a categorial grammar $\mathbf{G}$ to the cartesian closed category ΛD generated by the signature D which captures the language of *string diagrams*: boxes, identity, composition and tensor.

Before we draw any diagram, we fix a monoidal signature $\Sigma = (\Sigma_0, \Sigma_1, \text{dom}, \text{cod})$, i.e. a pair of sets Σ_0 and Σ_1 for generating *objects* and *boxes* together with a pair of functions $\text{dom}, \text{cod} : \Sigma_1 \rightarrow T$ from boxes to lists of objects $T = \Sigma_0^* = \coprod_{k \in \mathbb{N}} \Sigma_0^k$, also called *types*. The generating objects of the cartesian closed category ΛD are given by pairs of types which encode the input and output of diagrams, i.e. $D_0 = T \times T$.

D_1	symbol	type
boxes	$f \in \Sigma$	(x, y)
identity	id_x	(x, x)
composition	$\circ_{xyz}$	$(x, y) \rightarrow (y, z) \rightarrow (x, z)$
tensor	$\otimes_{xyzw}$	$(x, y) \rightarrow (z, w) \rightarrow (xz, yw)$

The generating arrows D_1 are listed in the table above, they are chosen so that the arrows:

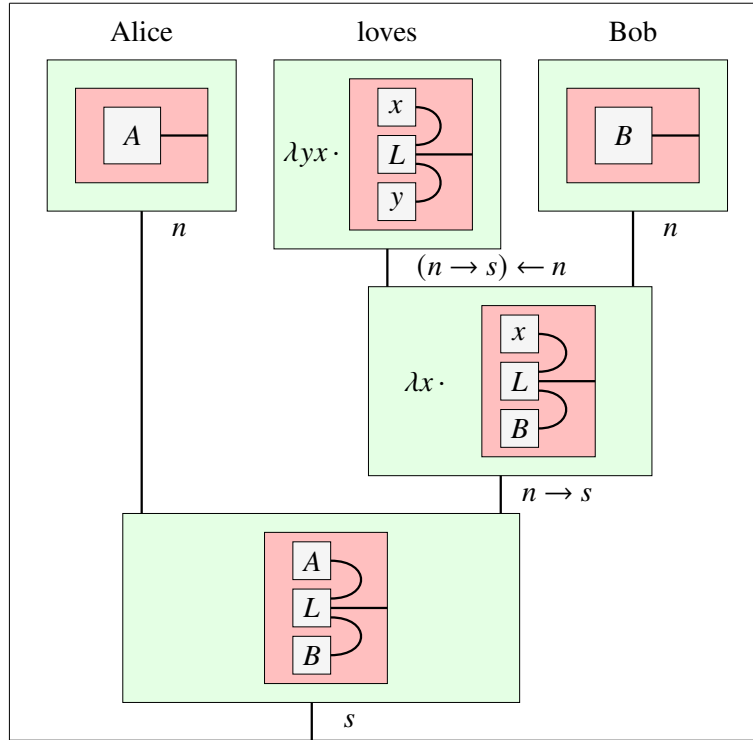
$$f : (z_1, w_1) \times \cdots \times (z_k, w_k) \rightarrow (x, y) \in \Lambda D$$

are all the well-typed string diagrams with $(x, y) \in D_0$ as domain, codomain and k free variables for boxes of shape $(z_i, w_i) \in D_0$. The relations $(f, f') \in D_2$ between pairs of diagrams are given by the axioms for monoidal categories so that $f = f'$ in ΛD if and only if the two diagrams are equivalent (i.e. planar isotopic), see [24, Theorem 3.1]. Closed monoidal functors $[[-]] : \Lambda D \rightarrow \mathbf{Set}$ are in one-to-one correspondence with monoidal functors $[[-]] : F(\Sigma) \rightarrow \mathbf{C}$ from the free monoidal category generated by Σ into any monoidal category $\mathbf{C}$. In other words, they are defined by the following data:

- an inclusion $\Sigma_0 \hookrightarrow \mathbf{C}_0$ and a morphism $[[f]] \in \mathbf{C}(x, y)$ for each box $(f : x \rightarrow y) \in \Sigma$,
- a *homset* $\mathbf{C}(x, y) = [[(x, y)]]$ between every pair of types,
- an *identity morphism* $[[\text{id}_x]] \in \mathbf{C}(x, x)$ for each type $x \in T$,
- a *composition function* $[[\circ_{xyz}]] : \mathbf{C}(x, y) \rightarrow \mathbf{C}(y, z) \rightarrow \mathbf{C}(x, z)$,
- a *tensor function* $[[\otimes_{xyzw}]] : \mathbf{C}(x, y) \rightarrow \mathbf{C}(z, w) \rightarrow \mathbf{C}(xz, yw)$.

We can recover the standard definition of DisCoCat models in terms of pregroups by taking the HO-DisCoCat model which sends the syntax trees in a categorial grammar to their corresponding pregroup diagrams, which come with an implicit interpretation in $\mathbf{C} = \mathbf{Vect}$. This is based on a functor from closed to *rigid monoidal categories* which appears in the CCG version of the DisCoCat framework [28].

CCG-to-DisCoCat	word w	type t	meaning $F(w \rightarrow t)$
proper nouns	Alice, Bob	n	$\text{Alice, Bob} \in \Sigma$
transitive verbs	loves	$(n \rightarrow s) \leftarrow n$	$\lambda f g. (\text{cup}_N \otimes \text{id}_S \otimes \text{cup}_N) \circ (g \otimes \text{loves} \otimes f)$



Remark. A similar trick of interpreting syntax trees as functions that return pregroup diagrams is also used in the formalisation of Wittgenstein’s language games in terms of functors [13].

We can also formalise our toy example “very big car” where we define the image of our HO-DisCoCat model $F : \mathbf{G} \rightarrow \Lambda D$ as follows.

“very big” = “big big”	word w	type t	meaning $F(w \rightarrow t)$
common nouns	car	n	$\text{car} \in \Sigma$
adjectives	big	$n \leftarrow n$	$\lambda x. \text{big} \circ x$
adverbs	very	$(n \leftarrow n) \leftarrow (n \leftarrow n)$	$\lambda f x. f f x$

Note that we have three different directions in which to compose diagrams:

1. *left-to-right* using the composition $\circ_{xyz} : (x, y) \times (y, z) \rightarrow (x, z)$ which connects the output of one diagram to the input of another,
2. *top-to-bottom* using the tensor $\otimes_{xyzw} : (x, y) \times (z, w) \rightarrow (xz, yw)$ which concatenates two diagrams side by side,
3. *inside-out* using the evaluation $(x, y) \times [(x, y) \rightarrow (z, w)] \rightarrow (z, w)$ which substitutes one diagram for the free variable in another.

Whereas standard DisCoCat model could play only with the first two directions, it is this extra direction of inside-out composition that makes our higher-order DisCoCat models more flexible. In the next section, we make this extra flexibility more explicit by adding it directly to the signature for our diagrams.

4 Adverbs as operators on homsets

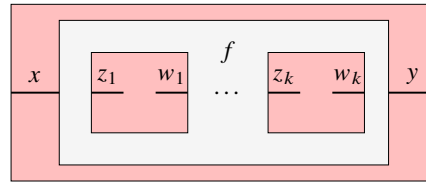
Moving away from our toy example to a more realistic model of language, we cannot assume that adverbs can all be given an explicit formula like the one we chose for “very”. We propose to represent such second-order³ processes as *boxes with k holes* that can contain diagrams of a certain shape. These processes take a first-order process (represented by the diagram inside the box) and return a new process (represented as the box itself). Concretely, we define a *monoidal signature with holes*:

$$\Sigma = (\Sigma_0, \Sigma_1, \text{dom}, \text{cod}, \text{holes})$$

as a monoidal signature together with a function $\text{holes} : \Sigma_1 \rightarrow (T \times T)^*$ which assigns to each box $(f : x \rightarrow y) \in \Sigma$ a list of pairs of types:

$$\text{holes}(f) = ((z_1, w_1), \dots, (z_k, w_k)) \in (T \times T)^*$$

for the domain and codomain of the diagrams that can fit in its $k \in \mathbb{N}$ holes. This is drawn as follows:



We may now add some dedicated symbols to our set of generators for the cartesian closed category ΛD , i.e. the primitives D_1 of our lambda calculus for diagrams.

D_1 (continued)	symbol	type
boxes with holes	$f \in \Sigma$	$(z_1, w_1) \rightarrow \dots \rightarrow (z_k, w_k) \rightarrow (x, y)$

The arrows $f : (z_1, w_1) \times \dots \times (z_k, w_k) \rightarrow (x, y)$ in ΛD are all the well-formed string diagrams where the boxes themselves can contain well-formed string diagrams, again with k boxes as free variables. The closed monoidal functors $[[-]] : \Lambda D \rightarrow \mathbf{Set}$ are precisely the monoidal categories $\mathbf{C}$ with a k -ary operator on homsets for each box $f \in \Sigma$:

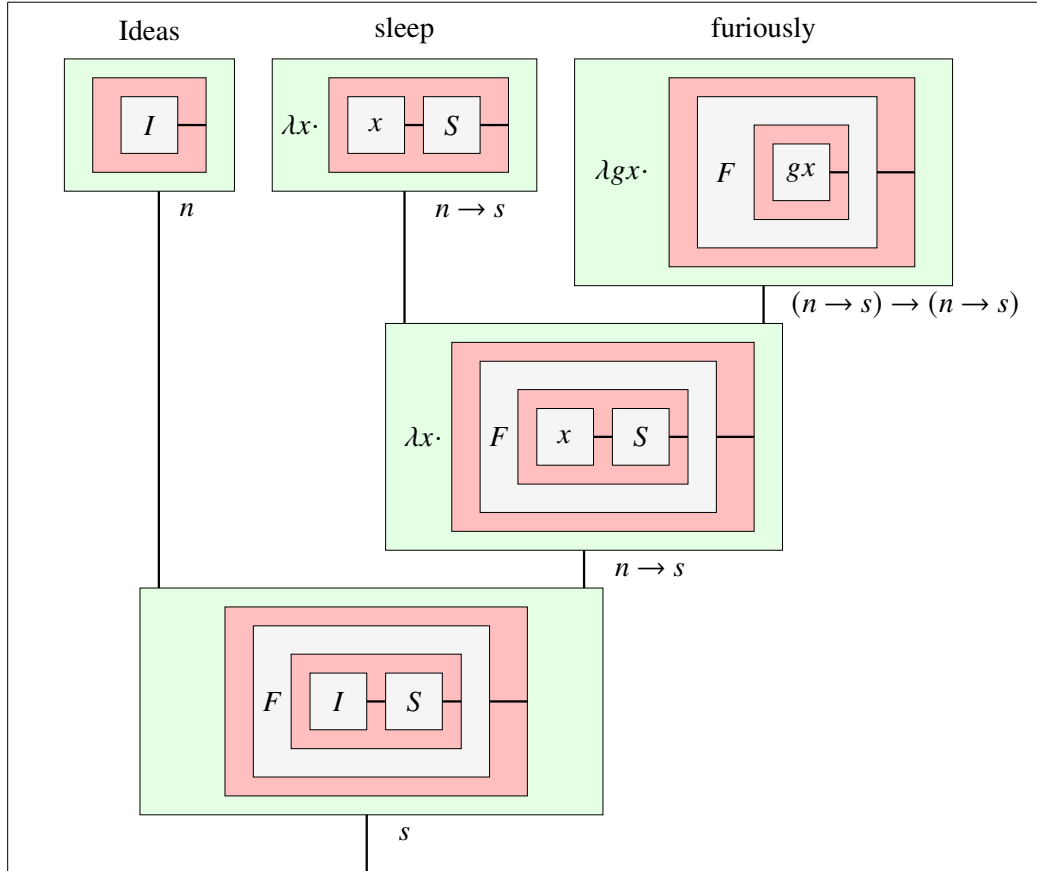
$$[[f]] : \mathbf{C}(z_1, w_1) \times \dots \times \mathbf{C}(z_k, w_k) \rightarrow \mathbf{C}(x, y)$$

Remark. A box with no holes is just a simple box; a nullary operator on homsets is just an element of a homset, i.e. a morphism. Boxes with one hole and unary operators on homsets have also been called bubbles, they were first introduced in [22] to denote the derivative of diagrams. As we will see in the next section, they also allow to formalise negation in Peirce’s system beta. Boxes with a list of holes first appeared under the name of quantum combs in [5]. These “open diagrams” have then been formalised in terms of the coend calculus [23, 16].

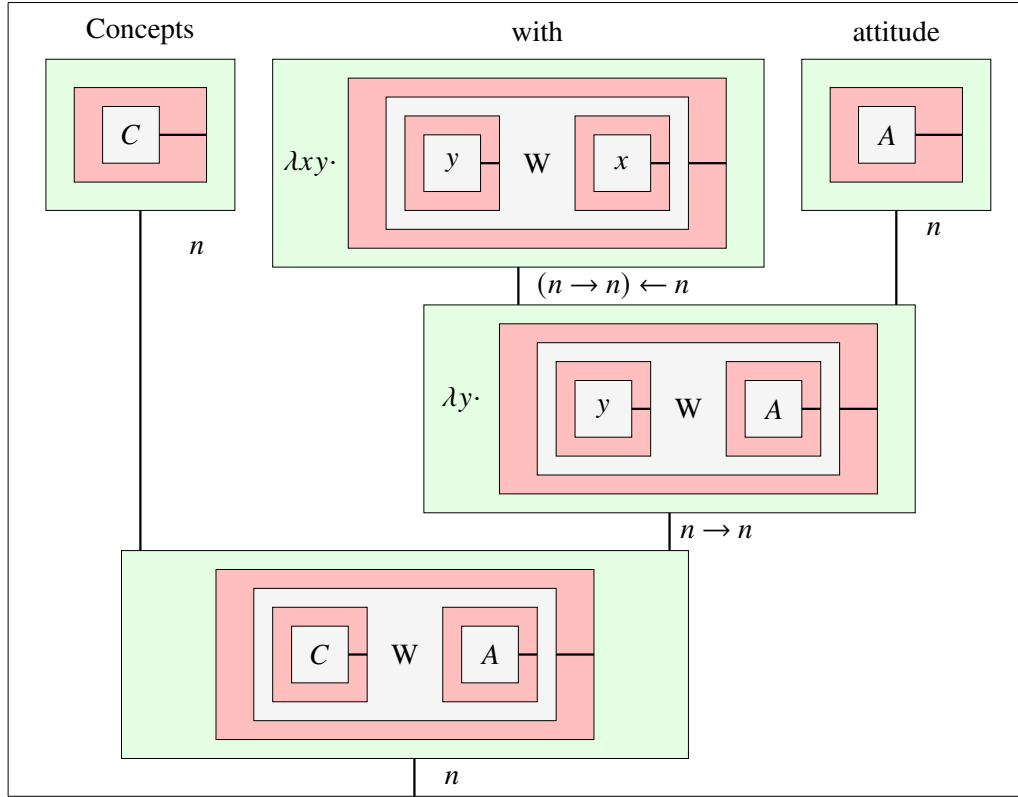
³We could imagine going higher still and consider boxes with holes that can fit diagrams with holes themselves, i.e. operators on operators on homsets. We stop at second-order because it is the limit of what we can draw with two-dimensional diagrams.

We can now define the semantics of adverbs like “furiously” as boxes with one hole for a verb, prepositions like “with” as boxes with two holes for nouns, etc.

HO-DisCoCat	word w	type t	meaning $F(w \rightarrow t)$
nouns	ideas	n	$I \in \Sigma$
intransitive verbs	sleep	$n \rightarrow s$	$\lambda x. S \circ x$
adverbs	furiously	$(n \rightarrow s) \rightarrow (n \rightarrow s)$	$\lambda f x. F(fx)$
prepositions	with	$(n \rightarrow n) \leftarrow n$	$\lambda x y. W(y, x)$



Remark. Note how in the following example “concepts with attitude”, we are merely interpreting the syntax tree as itself, represented in terms of nesting rather than branching. Indeed when a diagram has no composition or tensor, but only inside-out nesting of boxes with holes, it is nothing more than a syntax tree with operators on the nodes and homsets on the edges.



Remark. Composition and tensor can themselves be thought of as binary operators on homsets, subject to the axioms of monoidal categories: boxes with holes are all you need! This observation can be formalised in terms of operads of wiring diagrams [27].

5 Peirce-Lambek-Montague semantics

In generalising Montague semantics and DisCoCat models, we are in fact going back to their common ancestor: Peirce’s *existential graphs*, a formal graphical language of string diagrams for logic. Peirce’s existential graphs come in three variants of increasing expressive power:

- *System alpha*, the fragment of system beta with no wires and only scalar boxes, is equivalent to propositional logic and Boolean algebras.
- *System beta* is the variant we use in this work, it is equivalent to first-order logic.
- *System gamma* extends system beta with a second unary operator on homsets for the “contingent” modality (i.e. “not necessary”) together with axioms that make it sound and complete for various kinds of modal logics.

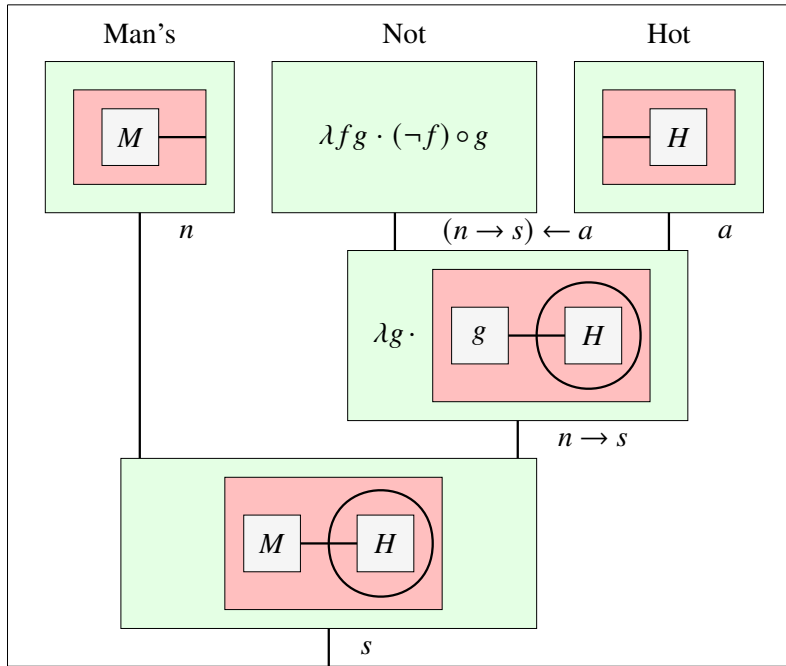
The translation between Peirce’s graphical notation and the traditional one is straightforward. Boxes are predicates and wires are variables, which are implicitly existentially quantified. Tensor is interpreted as conjunction and spiders as equality of variables. A circle surrounding a subdiagram is interpreted as negation of the corresponding subformula. Thus, a scalar diagram with no input or output wires

represents a first-order logic formula with no free variables, which we take as our interpretation for sentences. A diagram with no input and one output wire N is a formula with one free variable. We take this as our interpretation for common nouns, i.e.

$$F(s) = (1, 1) \quad \text{and} \quad F(n) = (1, N)$$

Before we move on to noun phrases and determiners, let us look at a preliminary example taken from Big Shaq's novelty song title “Man's Not Hot”, which we interpret as the formula $\exists x. \text{man}(x) \wedge \neg \text{hot}(x)$.

Man's Not Hot	word w	type t	meaning $F(w \rightarrow t)$
common noun	Man's	n	$(\text{man} : 1 \rightarrow N) \in \Sigma$
negation	Not	$(n \rightarrow s) \leftarrow a$	$\lambda f g. (\neg f) \circ g$
adjective	Hot	a	$(\text{hot} : N \rightarrow 1) \in \Sigma$



Here we had to define the type a for adjectives as atomic so that we could take $F(a) = (N, 1)$ instead of $F(n \rightarrow s) = (1, N) \rightarrow (1, 1)$. Indeed, the unary operator for negation can act only on diagrams, not diagram-valued functions. Suppose we had defined the meaning of “hot” in terms of post-composition $\lambda f. \text{hot} \circ f$ as in our previous example. Then the function for “not” would have had to extract the diagram $(N, 1)$ from the diagram-valued function $(1, N) \rightarrow (1, 1)$. The only way to do this would be to apply the function $\lambda x. \text{hot} \circ x$ to the identity diagram $\text{id}_N : (N, N)$, but that would be forbidden by its type. We need *dependent types* to express the fact that post-composition by the effect $\text{hot} : (N, 1)$ can take as input any diagram (x, N) and return a diagram of type $(x, 1)$.

This is precisely the definition we use to model noun phrases, i.e.

$$F(p) = \prod_x (N, x) \rightarrow (1, x)$$

For example, the function $\lambda f. \text{man} \circ f$ for the noun phrase “a man” can take as input any diagram (N, x) and return a state $(1, x)$. In this way, we can apply it to an effect $\text{hot} : (N, 1)$ and get a closed formula

$\exists x.\text{man}(x) \wedge \text{hot}(x)$, but we can also keep the variable free by applying it to the identity. Finally, we can define the meaning $\lambda f g.g(f(\text{id}_N)^T)$ of the copula “is”: it takes two noun phrases, applies one to the identity diagram and transposes the result before feeding it to the other. Other verbs are given in the same way, replacing the identity diagram id_N by any predicate. In the absence of a determiner, verbs can also be applied to common nouns by applying an ad-hoc rule $p \leftarrow n$, interpreted as pre-composition $F(p \leftarrow n) : (1, N) \rightarrow \prod_x (N, x) \rightarrow (1, x)$.

Peirce-Lambek-Montague	word w	type t	meaning $F(w \rightarrow t)$
common nouns	man	n	$(\text{man} : 1 \rightarrow N) \in \Sigma$
proper nouns	Alice	p	$\lambda f.\text{Alice} \circ f$
adjectives	big	$n \leftarrow n$	$\lambda f.\text{spider}_{2,1} \circ (\text{big} \otimes f)$
determiners	no	$p \leftarrow n$	$\lambda f g.\text{cut}(g \circ f)$
intransitive verbs	sleeps	$p \rightarrow s$	$\lambda P.P(\text{sleeps})$
transitive verbs	kills	$(p \rightarrow s) \leftarrow p$	$\lambda PQ.Q(P(\text{kills})^T)$
copula	is	$(p \rightarrow s) \leftarrow p$	$\lambda PQ.Q(P(\text{id}_N)^T)$

See Appendix A for an implementation of this functor $F : \mathbf{G} \rightarrow \Lambda D$ and three examples of sentences.

Conclusion

Higher-order DisCoCat models are a generalisation of DisCoCat and Montague semantics where the meaning of words is given by lambda terms which output string diagrams. We have given a formal definition in terms of functors $\mathbf{G} \rightarrow \Lambda D$ for $\mathbf{G}$ a categorial grammar and ΛD the cartesian closed category generated by the language of string diagrams. We defined a concrete instance of our model where the string diagrams are Peircean existential graphs: *Peirce-Lambek-Montague semantics*. Appendix A gives an implementation of these models as DisCoPy [12] functors into the category of Python functions.

A natural next step would be to give experimental support for our higher-order DisCoCat models. Another direction which has been developed recently [19] is to go from sentences to text, formalising the DisCoCirc framework [8] in terms of diagram-valued higher-order functions.

Acknowledgements The authors thank Sean Tull, Richie Yeung and Bob Coecke, as well as ACT reviewers 2 and 3 for their insightful support in improving this manuscript.

References

- [1] Samson Abramsky (2012): *No-Cloning In Categorical Quantum Mechanics*, doi:[10.48550/arXiv.0910.2401](https://doi.org/10.48550/arXiv.0910.2401), arXiv:[0910.2401](https://arxiv.org/abs/0910.2401).
- [2] Samson Abramsky & Nikos Tzevelekos (2011): *Introduction to Categories and Categorical Logic*. CoRR abs/1102.1313. arXiv:[1102.1313](https://arxiv.org/abs/1102.1313).
- [3] Johan Bos, Stephen Clark, Mark Steedman, James R. Curran & Julia Hockenmaier (2004): *Wide-Coverage Semantic Representations from a CCG Parser*. In: *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, COLING, Geneva, Switzerland, pp. 1240–1246. doi:[10.3115/1220355.1220535](https://doi.org/10.3115/1220355.1220535).
- [4] Geraldine Brady & Todd H. Trimble (1998): *A String Diagram Calculus for Predicate Logic and C. S. Peirce’s System Beta*. Available at <http://people.cs.uchicago.edu/~brady/beta98.ps>.

- [5] G. Chiribella, G. M. D’Ariano & P. Perinotti (2008): *Quantum Circuit Architecture*. *Phys. Rev. Lett.* 101(6), p. 060401, doi:[10.1103/PhysRevLett.101.060401](https://link.aps.org/doi/10.1103/PhysRevLett.101.060401). Available at <https://link.aps.org/doi/10.1103/PhysRevLett.101.060401>.
- [6] Stephen Clark, Bob Coecke & Mehrnoosh Sadrzadeh (2008): *A Compositional Distributional Model of Meaning*. In: *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pp. 133–140.
- [7] Stephen Clark, Bob Coecke & Mehrnoosh Sadrzadeh (2010): *Mathematical Foundations for a Compositional Distributional Model of Meaning*. In J. van Benthem & M. Moortgat, editors: *A Festschrift for Jim Lambek*, *Linguistic Analysis* 36, pp. 345–384. arXiv:[1003.4394](https://arxiv.org/abs/1003.4394).
- [8] Bob Coecke (2021): *The Mathematics of Text Structure*. In Claudia Casadio & Philip J. Scott, editors: *Joachim Lambek: The Interplay of Mathematics, Logic, and Linguistics*, Springer International Publishing, Cham, pp. 181–217, doi:[10.1007/978-3-030-66545-6_6](https://doi.org/10.1007/978-3-030-66545-6_6).
- [9] Bob Coecke, Giovanni de Felice, Konstantinos Meichanetzidis & Alexis Toumi (2020): *Foundations for Near-Term Quantum Natural Language Processing*. ArXiv e-prints. arXiv:[2012.03755](https://arxiv.org/abs/2012.03755).
- [10] Bob Coecke, Edward Grefenstette & Mehrnoosh Sadrzadeh (2013): *Lambek vs. Lambek: Functorial Vector Space Semantics and String Diagrams for Lambek Calculus*. *Ann. Pure Appl. Log.* 164(11), pp. 1079–1100, doi:[10.1016/j.apal.2013.05.009](https://doi.org/10.1016/j.apal.2013.05.009). arXiv:[1302.0393](https://arxiv.org/abs/1302.0393).
- [11] Giovanni de Felice (2022): *Categorical Tools for Natural Language Processing*. Ph.D. thesis, University of Oxford. arXiv:[2212.06636](https://arxiv.org/abs/2212.06636).
- [12] Giovanni de Felice, Alexis Toumi & Bob Coecke (2020): *DisCoPy: Monoidal Categories in Python*. In: *Proceedings of the 3rd Annual International Applied Category Theory Conference, ACT, 333, EPTCS*, doi:[10.4204/EPTCS.333.13](https://doi.org/10.4204/EPTCS.333.13).
- [13] Giovanni de Felice, Elena Di Lavore, Mario Román & Alexis Toumi (2020): *Functorial Language Games for Question Answering*. In David I. Spivak & Jamie Vicary, editors: *Proceedings of the 3rd Annual International Applied Category Theory Conference 2020, ACT 2020, Cambridge, USA, 6-10th July 2020, EPTCS 333*, pp. 311–321, doi:[10.4204/EPTCS.333.21](https://doi.org/10.4204/EPTCS.333.21).
- [14] Giovanni de Felice, Konstantinos Meichanetzidis & Alexis Toumi (2019): *Functorial Question Answering*. In: *Proceedings Applied Category Theory 2019, ACT 2019, University of Oxford, UK, EPTCS 323*, doi:[10.4204/EPTCS.323.6](https://doi.org/10.4204/EPTCS.323.6).
- [15] Nathan Haydon & Paweł Sobociński (2020): *Compositional Diagrammatic First-Order Logic*. In Ahti-Veikko Pietarinen, Peter Chapman, Leonie Bosveld-de Smet, Valeria Giardino, James Corter & Sven Linker, editors: *Diagrammatic Representation and Inference*, Lecture Notes in Computer Science, Springer International Publishing, Cham, pp. 402–418, doi:[10.1007/978-3-030-54249-8_32](https://doi.org/10.1007/978-3-030-54249-8_32).
- [16] James Hefford & Cole Comfort (2023): *Coend Optics for Quantum Combs*. *Electron. Proc. Theor. Comput. Sci.* 380, pp. 63–76, doi:[10.4204/EPTCS.380.4](https://doi.org/10.4204/EPTCS.380.4). arXiv:[2205.09027](https://arxiv.org/abs/2205.09027).
- [17] J. Lambek (1988): *Categorical and Categorical Grammars*. In Richard T. Oehrle, Emmon Bach & Deirdre Wheeler, editors: *Categorical Grammars and Natural Language Structures*, Studies in Linguistics and Philosophy, Springer Netherlands, Dordrecht, pp. 297–317, doi:[10.1007/978-94-015-6878-4_11](https://doi.org/10.1007/978-94-015-6878-4_11).
- [18] Joachim Lambek (1958): *The Mathematics of Sentence Structure*. *The American Mathematical Monthly* 65(3), pp. 154–170, doi:[10.1080/00029890.1958.11989160](https://doi.org/10.1080/00029890.1958.11989160).
- [19] Jonathon Liu, Razin A Shaikh, Benjamin Rodatz, Richie Yeung & Bob Coecke (2023): *Discourse Text Circuits From CCG: A Pipeline*. to appear.
- [20] Richard Montague (1970): *Universal Grammar*. *Theoria* 36(3), pp. 373–398, doi:[10.1111/j.1755-2567.1970.tb00434.x](https://doi.org/10.1111/j.1755-2567.1970.tb00434.x). Available at <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1755-2567.1970.tb00434.x>.
- [21] Charles Santiago Sanders Peirce (1906): *Prolegomena to an Apology of Pragmaticism*. *The Monist* 16(4), pp. 492–546. doi:[10.5840/monist190616436](https://doi.org/10.5840/monist190616436).

- [22] Roger Penrose & Wolfgang Rindler (1984): *Spinors and Space-Time: Volume 1: Two-Spinor Calculus and Relativistic Fields*. Cambridge Monographs on Mathematical Physics 1, Cambridge University Press, Cambridge, doi:[10.1017/CBO9780511564048](https://doi.org/10.1017/CBO9780511564048). Available at <https://www.cambridge.org/core/books/spinors-and-spacetime/B66766D4755F13B98F95D0EB6DF26526>.
- [23] Mario Román (2020): *Coend Calculus and Open Diagrams*. ArXiv e-prints. arXiv:[2004.04526](https://arxiv.org/abs/2004.04526).
- [24] Peter Selinger (2010): *A Survey of Graphical Languages for Monoidal Categories*. *New Structures for Physics*, pp. 289–355, doi:[10.1007/978-3-642-12821-9_4](https://doi.org/10.1007/978-3-642-12821-9_4).
- [25] Alexis Toumi (2022): *Category Theory for Quantum Natural Language Processing*. Ph.D. thesis, University of Oxford. arXiv:[2212.06615](https://arxiv.org/abs/2212.06615).
- [26] Jan van Eijck & Christina Unger (2010): *Computational Semantics with Functional Programming*. Cambridge University Press, Cambridge, doi:[10.1017/CBO9780511778377](https://doi.org/10.1017/CBO9780511778377). Available at <https://www.cambridge.org/core/books/computational-semantics-with-functional-programming/0D3BAC27C39751AE4FF7F08FCC1C1364>.
- [27] Donald Yau (2018): *Operads of Wiring Diagrams*. *Lecture Notes in Mathematics* 2192, Springer International Publishing, Cham, doi:[10.1007/978-3-319-95001-3](https://doi.org/10.1007/978-3-319-95001-3). Available at <http://link.springer.com/10.1007/978-3-319-95001-3>.
- [28] Richie Yeung & Dimitri Kartsaklis (2021): *A CCG-Based Version of the DisCoCat Framework*. ArXiv e-prints. arXiv:[2105.07720](https://arxiv.org/abs/2105.07720).

A Implementation and Examples

```

from random import choice
from typing import Callable

from discopy import frobenius, closed, python
from discopy.tensor import Dim, Tensor
from discopy.cat import Category, factory
from discopy.grammar.categorical import Ty, Word, Eval

# Step 1: Define Formula as a subclass of frobenius.Diagram

@factory
class Formula(frobenius.Diagram):
    ty_factory = frobenius.PRO # i.e. natural numbers as objects

    def eval(self, size):
        return frobenius.Functor(
            ob=lambda _: Dim(size),
            ar=lambda box: box.data,
            cod=Category(Dim, Tensor[bool]))(self)

class Cut(frobenius.Bubble, Formula): pass
class Ligature(frobenius.Spider, Formula): pass
class Predicate(frobenius.Box, Formula): pass

Id, Formula.bubble_factory = Formula.id, Cut
Tensor[bool].bubble = lambda self, **_: self.map(lambda x: not x)

# Step 2: Parse natural language sentences using a categorical grammar

n, p, s = Ty('n'), Ty('p'), Ty('s') # noun, phrase and sentence

```

```

Alice = Word("Alice", p)
big, sleeps = Word("big", n << n), Word("sleeps", p >> s)
man, island = (Word(noun, n) for noun in ("man", "island"))
kills, _is = (Word(verb, (p >> s) << p) for verb in ("kills", "is"))
no, every, some = (Word(det, p << n) for det in ("no", "every", "some"))

Alice_kills_a_mortal = (Alice @ kills @ some @ man
    >> p @ ((p >> s) << p) @ Eval(p << n)
    >> p @ Eval((p >> s) << p) >> Eval(p >> s))
every_big_man_sleeps = (every @ big @ man @ sleeps
    >> ((p << n) @ Eval(n << n) >> Eval(p << n))
    @ (p >> s) >> Eval(p >> s))
no_man_is_an_island = (no @ man @ _is @ some @ island
    >> Eval(p << n) @ ((p >> s) << p) @ Eval(p << n)
    >> p @ Eval((p >> s) << p) >> Eval(p >> s))

size = 42 # Generating a random interpretation to test our model
random_bits = lambda n=size: [choice([True, False]) for _ in range(n)]
is_killed_by = [random_bits() for _ in range(size)]
unary_predicates = is_Alice, is_man, is_island, is_big, is_sleeping = [
    random_bits() for _ in range(5)]

K = Predicate("K", 1, 1, data=is_killed_by)
A, M, I, B, S = (Predicate(P, 0, 1, data=
    for P, data in zip("AMIBS", unary_predicates))

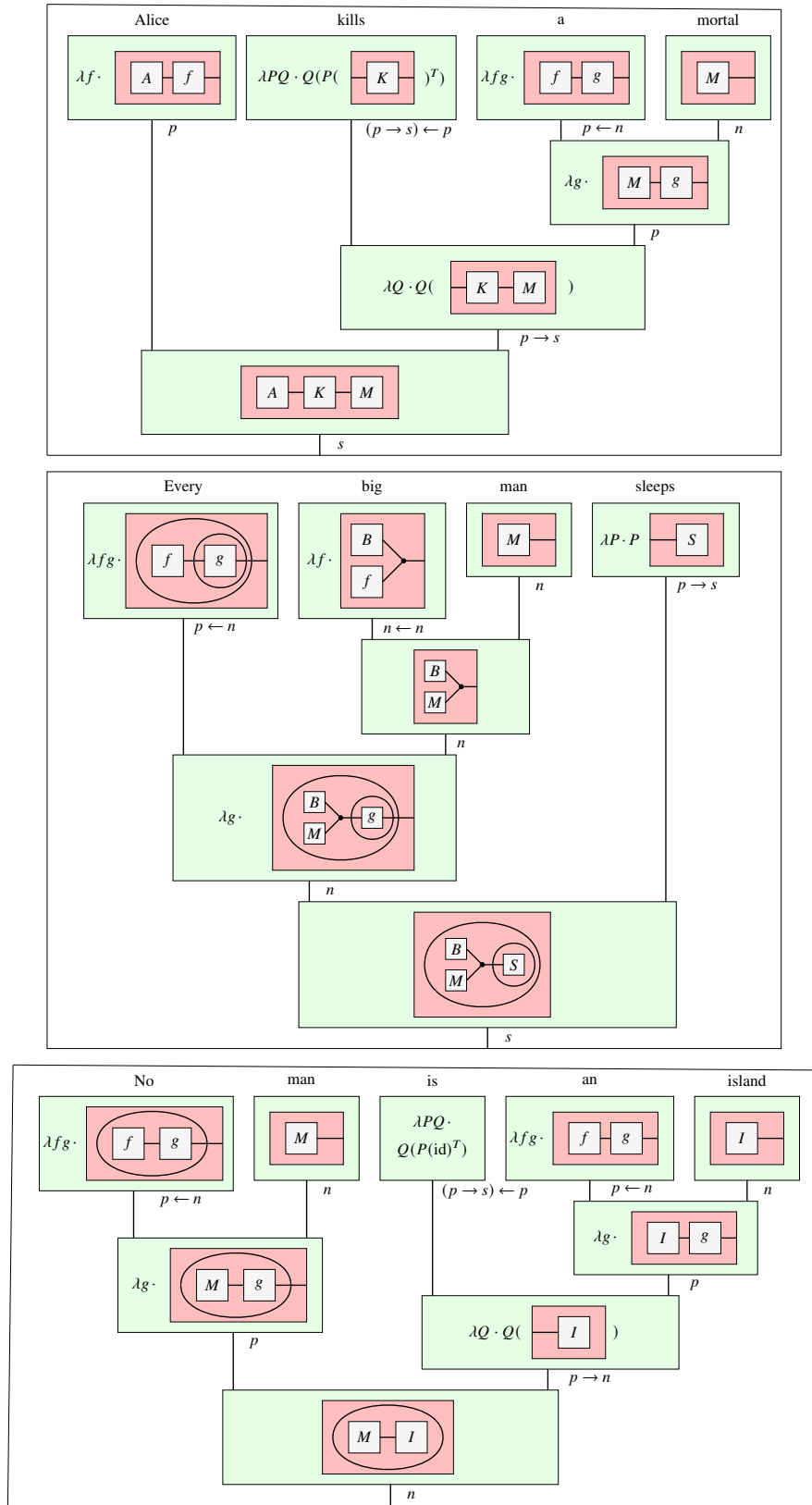
# Step 3: Higher-order DisCoCat as a closed functor into Python functions

F = closed.Functor(
    cod=Category(tuple[type, ...], python.Function),
    ob={s: Formula, n: Formula, p: Callable[[Formula], Formula]},
    ar={Alice: lambda: lambda f: A >> f,
        sleeps: lambda: lambda P: P(S.dagger()),
        man: lambda: M, island: lambda: I,
        big: lambda: lambda f: f @ B >> Ligature(2, 1, frobenius.PRO(1)),
        _is: lambda: lambda P: lambda Q: Q(P(Id(1)).dagger()),
        kills: lambda: lambda P: lambda Q: Q(P(K).dagger()),
        no: lambda: lambda f: lambda g: (f >> g).bubble(),
        some: lambda: lambda f: lambda g: f >> g,
        every: lambda: lambda f: lambda g: (f >> g.bubble()).bubble()})

evaluate = lambda sentence: bool(F(sentence()).eval(size))
assert evaluate(Alice_kills_a_mortal) == any(
    is_man[y] and is_killed_by[y][x] and is_Alice[x]
    for x in range(size) for y in range(size))
assert evaluate(every_big_man_sleeps) == all(
    not (is_big[x] and is_man[x]) or is_sleeping[x] for x in range(size))
assert evaluate(no_man_is_an_island) == all(
    not is_man[x] or not is_island[x] for x in range(size))

```

<https://docs.discopy.org/en/main/notebooks/higher-order-discocat.html>



Copy-composition for Probabilistic Graphical Models*

Toby St Clere Smithe

VERSES Research Lab

Topos Institute

act@tsmithe.net

In probabilistic modelling, joint distributions are often of more interest than their marginals, but the standard composition of stochastic channels is defined by marginalization. Recently, the notion of ‘copy-composition’ was introduced in order to circumvent this problem and express the chain rule of the relative entropy fibrationally, but while that goal was achieved, copy-composition lacked a satisfactory origin story. Here, we supply such a story for two standard probabilistic tools: directed and undirected graphical models. We explain that (directed) Bayesian networks may be understood as “stochastic terms” of product type, in which context copy-composition amounts to a pull-push operation. Likewise, we show that (undirected) factor graphs compose by copy-composition. In each case, our construction yields a double fibration of decorated (co)spans. Along the way, we introduce a useful bifibration of measure kernels, to provide semantics for the notion of stochastic term, which allows us to generalize probabilistic modelling from product to dependent types.

1 Introduction

Recent work has formalized the “chain rule” of the relative entropy (*a.k.a.* Kullback-Leibler divergence) as a section of a fibration of statistical games over Bayesian lenses [21]. Curiously, however, the base category of this fibration was not a category of stochastic channels (measure kernels), as one might expect, but instead a bicategory of *copy-composite* channels. The need for this stemmed from the Chapman-Kolmogorov rule for the composition of kernels, which involves marginalization; and marginalization does not commute with the logarithm defining the relative entropy. Copy-composition defers this marginalization (indeed, the Chapman-Kolmogorov rule is precisely the marginalization of the intermediate copy), so there is no problem of commutativity, and the chain rule thus obtains.

Although this yields an elegant fibrational structure, copy-composition is not strictly unital, owing to the extra copy produced when copy-composing with the identity. This means that the base of the fibration is necessarily a bicategory. Moreover, the construction of this bicategory [21, §2] was largely *ad hoc*, based on the similarity to a **CoPara** construction [1, Remark 4]. This was despite the evident importance of copy-composition to the statistical problems of interest there: copy-composition constructs *joint distributions*, which are often of more interest in applications than the composite channels that would yield their marginals. This can be seen not only via the relative entropy, but also in the context of Bayesian networks, which describe the factorization of a joint distribution according to a graphical structure. Fong [7] studied Bayesian networks from a categorical perspective, making extensive use of comonoid (‘copy-discard’ [2]) structure to construct the corresponding joint distributions; see for example the proof of his Theorem 4.5.

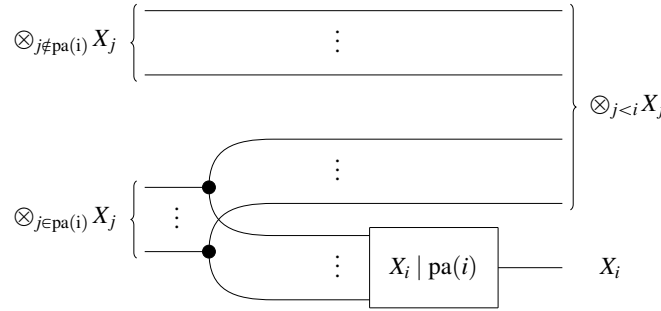
In the present submission, we show how copy-composition emerges naturally (but distinctly) from the compositional structure of directed and undirected probabilistic graphical models, thereby replacing

*Some of this content appeared first on the author’s website, such as at <https://tsmithe.net/p/factor-graphs.html>; however, none has appeared elsewhere.

the earlier arbitrariness with something structurally meaningful. In the directed case, the key observation is that a Bayesian network may be understood as a “stochastic term” of some displayed type (which is invariably a product), equipped with a distribution (the ‘prior’) over its context. Identifying types with bundles and terms with sections, this means understanding the factors of a Bayesian network as *stochastic sections*; copy-composition then emerges via a pull-push operation. In the undirected case, one observes that the ‘factors’ of a factor graph are costates¹ in a copy-discard category, and that copy-composition amounts to a kind of gluing. In both cases, copy-composition appears as the horizontal composition operation of a pseudo double category, thereby extending the earlier bicategorical story. These double categories² are in fact double fibrations, following the generalized decorated (co)span construction of Patterson [18]: in the directed case, one decorates spans of display maps with sections; in the undirected case, one decorates cospans with factors.

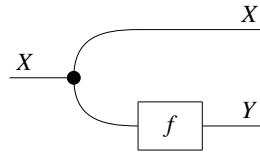
2 Directed models: pull-push stochastic sections

A *Bayesian network* is a distribution (or measure) on a product space that factorizes into a product of conditional distributions (or measure kernels) that matches the structure of a directed acyclic graph [7, Definition 3.7]. Interpreting this idea in a category of stochastic channels (such as measure kernels between measurable spaces), Fong [7, Theorem 4.5] showed that any Bayesian network can be written as a state $1 \rightarrow \otimes_j X_j$ that factorizes as the sequential composition of morphisms of the form



where $\text{pa}(i)$ denotes the set of parents of the vertex i , and where $j < i$ iff there is no directed path from i to j (Fong calls this the ‘ancestral’ order on the vertices).

Each factor $X_i | \text{pa}(i)$ only appears in the composite diagram as precomposed by a copier applied to its parents, and tensored with the identities of its grandparents. Interpreting the analogous string diagram



in **Set**, we see that it corresponds to the graph of the function $f : X \rightarrow Y$, which is a section of the product projection $X \times Y \rightarrow X$; in other words, a term of $X \times Y$ over the context X . Moreover, ‘tensoring’ this diagram with extra objects W corresponds to pulling back the section along the projection $W \times X \rightarrow X$;

¹A *state* in a monoidal category is a morphism out of the monoidal unit; a *costate* is a morphism into the monoidal unit.

²We take “double category” henceforth to mean *pseudo double category*, oriented such that composition is strict in the vertical category and weak in the horizontal.

in other words, extending the context by W . Thus, whereas Fong [7] constructed the preceding diagram by hand, with the right categorical setting we will be able to interpret it as resulting from standard type-theoretic operations. Our first task will therefore be to establish a bifibration of measure kernels in which these operations may be interpreted. Then, we will construct a double category whose horizontal composition implements these operations (and thus copy-composition) via ‘pull-push’ in the bifibration.

2.1 A bifibration of quasi-Borel kernels

The classic categorical semantics for dependent type theory is in fibrations [12, 17]: the base category is interpreted as a category of ‘contexts’; the fibre over an object interprets the types in the corresponding context; elements in a fibre interpret terms; and pullback (base change) interprets substitution. The classic example is the codomain fibration of a finitely complete category, in which bundles ‘display’ types, and their sections are corresponding terms. The category **Meas** of measurable spaces and measurable functions between them is a finitely complete category, and so this story may be told there. But measurable functions are still deterministic; they are not kernels.

One may hope to replace the functions in the fibres by kernels and still obtain a fibration, but the image of a measurable subset under an arbitrary measurable map may not again be measurable³, which means that we cannot generally pull back traditional kernels. Still, all is not lost, for we may turn our attention to quasi-Borel spaces [11], where the basic notion of *measurable subset* is replaced by that of *random element*, and the notions of measure and kernel are adjusted accordingly. Additionally, there is an adjunction between **Meas** and the category **QBS** of quasi-Borel spaces that restricts to an equivalence on standard Borel spaces [11, Prop. 15], so we do not stray too far from the familiar.

Definition 2.1 (QBS). Let $\mathbb{I}$ denote the unit interval $[0, 1]$. A *quasi-Borel space* is a pair (X, M_X) of a set X along with a subset $M_X \subseteq \mathbf{Set}(\mathbb{I}, X)$ of functions $\mathbb{I} \rightarrow X$ (*random elements* of X), satisfying the following axioms:

1. (*constants*) if $\rho : \mathbb{I} \rightarrow X$ is a constant function, then $\rho \in M_X$;
2. (*closure under measurable precomposition*) if $f : \mathbb{I} \rightarrow \mathbb{I}$ is measurable (i.e., $f \in \mathbf{Meas}(\mathbb{I}, \mathbb{I})$) and $\rho \in M_X$, then $\rho \circ f \in M_X$;
3. (*sheaf*) if $\mathbb{I}$ is partitioned as $\mathbb{I} = \coprod_{i \in \mathbb{N}} S_i$ with each $S_i \in \Sigma_{\mathbb{I}}$, and if, for each i , we have $\alpha_i \in M_X$, then $\coprod_{i \in \mathbb{N}} \alpha_i \in M_X$.

A morphism of quasi-Borel spaces $f : (X, M_X) \rightarrow (Y, M_Y)$ is a function $f \in \mathbf{Set}(X, Y)$ such that $\rho \in M_X$ entails $f \circ \rho \in M_Y$. These morphisms compose as functions, yielding a category, **QBS**.

Remark 2.2. The definition of quasi-Borel space given by Heunen et al. [11] uses $\mathbb{R}$ as the source of randomness, rather than $\mathbb{I}$. But $\mathbb{R}$ and $\mathbb{I}$ equipped with their standard Borel structure are measurably isomorphic, so nothing is lost by this change, and it simplifies the presentation that follows.

Any quasi-Borel space (X, M_X) can be made into a measurable space (X, Σ_X) whose σ -algebra is given by $\Sigma_X = \{U \subseteq X \mid \forall \alpha \in M_X. \alpha^{-1}(U) \in \Sigma_{\mathbb{I}}\}$, where $\Sigma_{\mathbb{I}}$ is the Borel σ -algebra associated to $\mathbb{I}$; this is Prop. 14 of Heunen et al. [11]. By equipping the noise source $\mathbb{I}$ with a measure, the random elements become random variables. We can then push forward the noise to obtain measures on (the measurable space associated to) each quasi-Borel space. Thus we can consider the set of probability measures on a quasi-Borel space to be the set of measures induced in this way.

³This is guaranteed when the map is a continuous injection, by the Lusin-Souslin theorem [14, Theorem 15.1], but not in general. Thanks to Ohad Kammar for pointing this out to me.

Definition 2.3. Let ν denote the uniform measure on $\mathbb{I}$. A *probability measure* on a quasi-Borel space (X, M_X) is an equivalence class of elements of M_X , quotiented by equivalence of measure under pushforward of ν . Let $P(X)$ denote the set of such measures on (X, M_X) , which we can write as $\{\alpha_* \nu \in G(X, \Sigma_X) \mid \alpha \in M_X\}$, where $\alpha_* \nu$ denotes the pushforward measure $U \mapsto \nu(\alpha^{-1}(U))$. It has a quasi-Borel structure given by $M_{P(X)} = \{\beta : \mathbb{I} \rightarrow P(X) \mid \exists \alpha : \mathbb{I} \rightarrow M_X. \forall r \in \mathbb{I}. \beta(r) = \alpha(r)_* \nu\}$.

P is then made into a functor $\mathbf{QBS} \rightarrow \mathbf{QBS}$ by $P(f) : \alpha_* \nu \mapsto (f \circ \alpha)_* \nu$, and finally upgraded to a monad by analogy with the Giry monad $G : \mathbf{Meas} \rightarrow \mathbf{Meas}$; this is the content of §V.D of Heunen et al. [11]. (Equivalently, P is a quotient of the $\mathbb{I}$ -continuation monad by equivalence of measure.) A kernel between quasi-Borel spaces is then a morphism in the Kleisli category $\mathcal{Kl}(P)$, which we denote with squiggly arrows. The standard inclusion of $\mathbf{QBS}$ into $\mathcal{Kl}(P)$, given by post-composing with the monad unit, maps each function $f \in \mathbf{QBS}(X, Y)$ to a deterministic kernel $\delta_f : X \rightsquigarrow Y$.

We will use $\mathcal{Kl}(P)$ to define a bifibration $\mathcal{K}$ over $\mathbf{QBS}$, such that the objects of each fibre $\mathcal{K}_B$ will be (deterministic) functions into B , and the morphisms $(E, \pi) \rightsquigarrow (E', \pi')$ in $\mathcal{K}_B$ will be kernels $E \rightsquigarrow E'$ “fibrewise over B ”.

Definition 2.4 (The fibres of $\mathcal{K}$). Let B be a quasi-Borel space; we define a corresponding category $\mathcal{K}_B$. Its objects are pairs (E, p) of a quasi-Borel space E and a quasi-Borel function $p : E \rightarrow B$. A morphism $k : (E, p) \rightarrow (E', p')$ is a kernel $k : E \rightsquigarrow E'$ such that $\delta_{p'} \circ k = \delta_p$; we call these morphisms *kernels over B* . Composition of morphisms is given by the Kleisli composition of the corresponding kernels; identities are given by identity kernels. (It is easy to see that this yields a well-defined category.)

To see that the morphisms of $\mathcal{K}_B$ really are kernels fibrewise over B , let us introduce some notation.

Notation 2.5 (Fibre notation). Suppose $p : E \rightarrow B$ is a function and $b : J \rightarrow B$ is a generalized element of B (another function). The fibre of p over b is given by the pullback of p along b :

$$\begin{array}{ccc} p[b] & \xrightarrow{\pi_E} & E \\ b^*p \downarrow & \lrcorner & \downarrow p \\ J & \xrightarrow{b} & B \end{array}$$

We denote the fibre by $p[b] \xrightarrow{b^*p} J$.

Proposition 2.6 (Morphisms in $\mathcal{K}_B$ are fibrewise kernels). Suppose $k : (E, p) \rightsquigarrow (E', p')$ is a kernel over B , and $b : J \rightarrow B$ is a generalized element of B . Then k restricts to a kernel $k[b] : p[b] \rightsquigarrow p'[b]$.

Proof. δ_p maps each $x \in E$ to the equivalence class generated by the constant random element $r \mapsto p(x)$. So the condition $\delta_{p'} \circ k = \delta_p$ means that $p(x) = p'(\overline{k(x)}(r))$, for all $x \in E$, $r \in \mathbb{I}$, and elements $\overline{k(x)}$ of the equivalence class corresponding to $k(x)$. Let $\pi_E : p[b] \rightarrow E$ be the projection as in the preceding pullback square; and define $k[b]$ by mapping $x \in p[b]$ to $k(\pi_E(x))$. Let $j = b^*p(x)$. By assumption, we have $p(\pi_E(x)) = b(j)$. We need to verify that for all $r \in \mathbb{I}$, $\overline{k(\pi_E(x))}(r) \in p'[b]$. By definition, we have $p'(\overline{k(\pi_E(x))}(r)) = p(\pi_E(x)) = b(j)$. So $\overline{k(\pi_E(x))}(r)$ is in the fibre of p' over $b(j)$, which is contained within $p'[b]$. So $\overline{k(\pi_E(x))}(r) \in p'[b]$ as required. $\square$

It will be very useful to extend this fibrewise-restriction to composite kernels; fortunately, it commutes with composition. For the proof, see §A.1.

Proposition 2.7 (Fibrewise restriction commutes with composition). $(k \circ h)[b] = k[b] \circ h[b]$, whenever $k \circ h$ exists.

It is moreover easy to check that fibrewise restriction preserves identity kernels, which, together with the preceding proposition, means that it is functorial. In particular, given $b : J \rightarrow B$ in **QBS**, fibrewise restriction yields a reindexing functor $\Delta_b : \mathcal{K}_B \rightarrow \mathcal{K}_J$.

Corollary 2.8 (Substitution). Suppose $b : J \rightarrow B$ in **QBS**. Then we have a functor $\Delta_b : \mathcal{K}_B \rightarrow \mathcal{K}_J$ as follows. On objects, Δ_b acts to map $E \xrightarrow{p} B$ to $p[b] \xrightarrow{b^*p} J$. On morphisms, it maps k to $k[b]$.

Proposition 2.9 (The stochastic self-indexing, and resulting fibration). Together with the substitution functors Δ , the mapping $\mathcal{K} : B \mapsto \mathcal{K}_B$ defines a pseudo functor $\mathbf{QBS}^{\text{op}} \rightarrow \mathbf{Cat}$. Pseudo functoriality follows from the (pseudo) functoriality of pullback and the functoriality of composition. Applying the Grothendieck construction to this indexed category yields a fibration $\int \mathcal{K} \rightarrow \mathbf{QBS}$.

It remains to exhibit left adjoints to the substitution functors Δ_b .

Proposition 2.10 (Dependent sum functor). Suppose $f : C \rightarrow B$ is a morphism in **QBS**. Then there is a functor $\Sigma_f : \mathcal{K}_C \rightarrow \mathcal{K}_B$ defined on objects $E \xrightarrow{p} C$ by post-composition, $\Sigma_f(p) := E \xrightarrow{p} C \xrightarrow{f} B$ and on kernels as the identity.

Note that, fibrewise, we have $\Sigma_f(p)[b] \cong p[f^*b]$. Since f^* is pullback in **QBS** and Δ is defined by pullback, this is the first sign of the adjointness of Σ and Δ . See §A.1 for the proof of the proposition:

Proposition 2.11 ($\Sigma \dashv \Delta$). Suppose $f : B \rightarrow C$ in **QBS**. Then Σ_f is left adjoint to Δ_f .

Thus $\mathcal{K}$ yields a bifibration. But this is not all we need: for copy-composition to be well-defined via pull-push, we need the functors Σ and Δ to satisfy the Beck-Chevalley condition. This result follows because the self-indexing of **QBS** satisfies Beck-Chevalley; but an explicit proof may be found in §A.1.

Proposition 2.12 ($\mathcal{K}$ satisfies Beck-Chevalley). Suppose we have a pullback square in **QBS** of the form

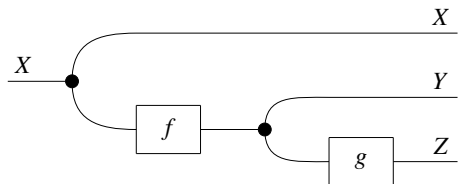
$$\begin{array}{ccc} & P & \\ \pi \swarrow & \downarrow & \searrow \rho \\ E & & F \\ p \searrow & \swarrow q & \\ & B & \end{array} .$$

Then there is a natural isomorphism $\Sigma_p \Delta_\pi \cong \Delta_q \Sigma_p$.

Remark 2.13. In fact, the bifibration $\int \mathcal{K}$ has more structure than we have space to explore in the present study: each fibre has a tensor product that makes it into a monoidal fibration; additionally, if we extend to *s-finite* kernels [22], each fibre has biproducts (and, with an inconsequential modification, a zero object); and thus the subcategory of probability kernels in each fibre is a *commutative effectus* [3]. Moreover, these structures interact nicely with the base change functors. We leave the exposition of all this to future work.

2.2 Decorating spans with sections

Our task now is to define a categorical structure in which morphisms include graphs (in the functional sense) and compose by copy-composition. That is, given the ‘graphs’ of $f : X \rightarrow Y$ and $g : Y \rightarrow Z$ in some underlying monoidal category, we wish to be able to compose them to yield



This composite is a section of the composite projection $X \times Y \times Z \rightarrow X \times Y \rightarrow X$. In turn, we may observe that this projection is the left leg of the composite span

$$\begin{array}{ccccc}
 & & X \times Y \times Z & & \\
 & \swarrow & \downarrow & \searrow & \\
 & X \times Y & & Y \times Z & \\
 \swarrow \pi_X^{XY} & & \pi_Y^{XY} & \pi_Y^{YZ} & \searrow \pi_Z^{YZ} \\
 X & & Y & & Z
 \end{array}$$

and that the graphs of f and g are sections of the left legs of the constituent spans, respectively. Thus, if we can interpret $\text{graph}(f)$ and $\text{graph}(g)$ bifibrationally, we could write their composite (depicted above) as $\Sigma_{\pi_X^{XY}} \Delta_{\pi_Y^{XY}} (\text{graph}(g)) \circ \text{graph}(f)$. The operation $\Sigma_{\pi_X^{XY}} \Delta_{\pi_Y^{XY}}$ is known as a “pull-push” operation [19], because one first pulls back along π_Y^{XY} and then pushes forward along π_X^{XY} .

When f and g are morphisms in a finitely complete category, we can immediately make sense of this pull-push composition in the codomain fibration. Likewise, when they are s-finite kernels, we can interpret this story in $\mathcal{K}$, and this is the situation that allows us to construct Bayesian networks and recover the chain rule of the relative entropy. Nonetheless, all we require is a bifibration satisfying Beck-Chevalley (and a section of it). The idea is then to decorate spans in the base of the bifibration with sections of the left leg, and then to compose them by pullback on the spans and pull-push on the decorations. Thus we have a “decorated span” construction, in the sense of Fong [8] and Patterson [18].

Patterson [18] observed that adding compositional decorations to categorical constructions often signals a Grothendieck fibration. Since spans (and cospans) collect into double categories, the decorated (co)span construction can be reconceived as a double Grothendieck construction, giving an extra dimension of composition to that originally proposed by Fong [8], and newly allowing the decorations to depend on whole (co)spans (rather than merely their apices), a freedom we will need. Following Cruttwell et al. [5, Theorem 3.51], the double Grothendieck construction produces a double fibration from an indexed double category, *i.e.* a lax double pseudo functor whose codomain is the double 2-category $\mathbf{Span}(\mathbf{Cat})$ of spans of categories; see Patterson [18, §3.1] for a summary.

Definition 2.14. A lax double pseudo functor $F : \mathbb{C} \rightarrow \mathbb{D}$ consists of a pair of pseudo functors $F_0 : \mathbb{C}_0 \rightarrow \mathbb{D}_0$ (from the vertical category of $\mathbb{C}$ to that of $\mathbb{D}$) and $F_1 : \mathbb{C}_1 \rightarrow \mathbb{D}_1$ (from the horizontal category of $\mathbb{C}$ to that of $\mathbb{D}$), along with pseudo natural transformations μ and η witnessing the compatibility of F_1 and F_0 with the double category structures of $\mathbb{C}$ and $\mathbb{D}$, as in the diagrams

$$\begin{array}{ccc}
 \mathbb{C}_1 \times_{\mathbb{C}_0} \mathbb{C}_1 & \xrightarrow{\diamond_{\mathbb{C}}} & \mathbb{C}_1 \\
 F_1 \times F_1 \downarrow & \nearrow \mu & \downarrow F_1 \\
 \mathbb{D}_1 \times_{\mathbb{D}_0} \mathbb{D}_1 & \xrightarrow{\diamond_{\mathbb{D}}} & \mathbb{D}_1
 \end{array}
 \qquad
 \begin{array}{ccc}
 \mathbb{C}_0 & \xrightarrow{I_{\mathbb{C}}} & \mathbb{C}_1 \\
 F_0 \downarrow & \nearrow \eta & \downarrow F_1 \\
 \mathbb{D}_0 & \xrightarrow{I_{\mathbb{D}}} & \mathbb{D}_1
 \end{array}$$

such that four standard coherence axioms [5, Def. 3.12] are satisfied. ($\diamond$ denotes the ‘external’ (2-cell) composition of a double category structure and I its unit, so that, in the context of indexed double categories, μ implements the *indexed* external composition, and η implements *its* unit.)

An *indexed double category* is a lax double pseudo functor with codomain $\mathbf{Span}(\mathbf{Cat})$.

Suppose now that $\mathcal{E} \xrightarrow{\pi} \mathcal{B}$ is a bifibration satisfying Beck-Chevalley, equipped with a section $\iota : \mathcal{B} \rightarrow \mathcal{E}$, where $\mathcal{B}$ has all pullbacks; we will continue to write $\mathcal{E}_B$ for the fibre over $B : \mathcal{B}$, and $\Sigma \dashv \Delta$ for the base change functors, and assume that Δ preserves ι . Our model for $\mathcal{E}$ will be either $\mathcal{K}$ or

the codomain fibration of a finitely complete category; and for ι the mapping $B \mapsto (\text{id}_B : B \rightarrow B)$, so that morphisms out of ι are understood as sections (and thus models for terms). We will construct an indexed double category $S : \mathbf{Span}(\mathcal{B})_{pb}^{\text{op}} \rightarrow \mathbf{Span}(\mathbf{Cat})$, where $\mathbf{Span}(\mathcal{B})_{pb}$ is the sub-double category of $\mathbf{Span}(\mathcal{B})$ whose 2-cells are Cartesian morphisms of spans (a restriction necessary for morphisms of decorations to be well-defined), and the $^{\text{op}}$ is taken vertically.

The vertical category of $\mathbf{Span}(\mathcal{B})_{pb}$ is $\mathcal{B}$, and the vertical 2-category of $\mathbf{Span}(\mathbf{Cat})$ is $\mathbf{Cat}$; but the vertical decorations will be trivial: S_0 is simply the constant functor on the terminal category $\mathbf{1}$.

There is a simplification in the horizontal direction, too. The horizontal category of $\mathbf{Span}(\mathcal{B})_{pb}$ is $\mathcal{B}_{pb}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}}$, the category of diagrams in $\mathcal{B}$ of shape $\bullet \leftarrow \bullet \rightarrow \bullet$, with morphisms restricted to those natural transformations that are Cartesian (*i.e.*, constituting pullback squares). S_1 must map a span $m = (A \xleftarrow{a} E \xrightarrow{b} B)$ in $\mathcal{B}$ to a span $\mathbf{1} \leftarrow S_1(m) \rightarrow \mathbf{1}$ of categories. Because $\mathbf{1}$ is terminal in $\mathbf{Cat}$, the legs of the latter span are accordingly determined. There is a further simplification: $S_1(m)$ will be a discrete category (a set). Indeed, we define $S_1(m)$ to be the set of sections of the left leg of m in $\mathcal{E}$. That is, $S_1(m) := \mathcal{E}_A(\iota A, \Sigma_a \iota E)$.

Given a morphism $f : m \rightarrow m'$ in $\mathbf{Span}(\mathcal{B})_{pb}$ as in the diagram

$$\begin{array}{ccccc} A & \xleftarrow{a} & E & \xrightarrow{b} & B \\ f_l \downarrow & & \downarrow & & \downarrow f_r \\ A' & \xleftarrow{a'} & E' & \xrightarrow{b'} & B' \end{array},$$

$S_1(f)$ must be a function $\mathcal{E}_{A'}(\iota A', \Sigma_{a'} \iota E') \rightarrow \mathcal{E}_A(\iota A, \Sigma_a \iota E)$, which we take to be given by Δ_{f_l} . This is well defined because $\Delta_{f_l} \iota A' = \iota_A \text{ ex hypothesis}$, $\Delta_{f_l} \Sigma_{a'} \iota E' \cong \Sigma_a \Delta_f \iota E'$ by Beck-Chevalley, and $\Delta_f \iota E' = \iota E \text{ ex hypothesis}$. Moreover, since Δ_{f_l} is functorial, so is $S_1(f)$.

The copy-composition of sections is implemented by the transformation μ , whose component at a pair of composable spans $m = (A \xleftarrow{a} E \xrightarrow{b} B)$ and $n = (B \xleftarrow{b'} F \xrightarrow{c} C)$ must be a function $\mu_{m,n} : S_1(m) \times S_1(n) \rightarrow S_1(n \diamond m)$, which we take to be defined by pull-push. That is, $\mu_{m,n}(\sigma, \tau) := \Sigma_a \Delta_b(\tau) \circ \sigma$. Associativity of μ follows by Beck-Chevalley; the explicit proof is in Appendix A.2. The unit η is defined over each identity span $E = E = E$ by the function $\eta_E : \mathbf{1} \rightarrow \mathcal{E}_E(\iota E, \iota E)$ determined by the identity on ι_E . Note that this makes pull-push copy-composition strictly unital, unlike in St Clere Smithe [21].

Applying the double Grothendieck construction [5, Theorem 3.51] to S yields a double category $\mathbb{S}$, doubly fibred over $\mathbf{Span}(\mathcal{B})_{pb}$, which has the following structure:

1. The objects of $\mathbb{S}$ are the objects of $\mathcal{B}$.
2. A vertical 1-cell is a morphism of $\mathcal{B}$, and vertical composition is as in $\mathcal{B}$.
3. A horizontal 1-cell $A \rightarrowtail B$ is a quadruple (E, a, b, σ) where $(A \xleftarrow{a} E \xrightarrow{b} B)$ is a span with σ being a section of the left leg in $\mathcal{E}_A$; *i.e.*, $\sigma \in \mathcal{E}_A(\iota A, \Sigma_a \iota E)$.
4. A 2 cell from $(E, a, b, \sigma) : A \rightarrowtail B$ to $(E', a', b', \sigma') : A' \rightarrowtail B'$ is a Cartesian morphism of spans $(f_l, f, f_r) : (A \xleftarrow{a} E \xrightarrow{b} B) \Rightarrow (A' \xleftarrow{a'} E' \xrightarrow{b'} B')$ such that $\sigma = \Delta_{f_l} \sigma'$.
5. Horizontal composition is by pullback on the spans (and their morphisms) and by pull-push on the sections, as described above.

That is to say, we have the following theorem (given the details in Appendix A.2).

Theorem 2.15 (Copy-composition of sections). Given a bifibration $\mathcal{E} \rightarrow \mathcal{B}$ over a finitely complete base, satisfying Beck-Chevalley, and equipped with a section $\iota : \mathcal{B} \rightarrow \mathcal{E}$ preserved by pullback, the foregoing data define a pseudo double category $\mathbb{S}$ doubly fibred over $\mathbf{Span}(\mathcal{B})_{pb}$.

By instantiating $\mathbb{S}$ in $\int \mathcal{K}$, we can formally recover the factors of a Bayesian network described by Fong [7] and introduced at the beginning of this section. There is a lax embedding of **sfKrn** horizontally into $\mathbb{S}$ (so instantiated), which maps a kernel to its graph. Let σ denote the image of the kernel $X_i \mid \text{pa}(i)$ under this embedding, which has the effect of precomposing it with copiers. Then, to tensor with the identity wires $\otimes_{j \notin \text{pa}(i)} X_j$, we can pull σ back along the projection $\prod_{j < i} X_j \rightarrow \prod_{j \in \text{pa}(i)} X_j$, as in the diagram

$$\begin{array}{ccccc} \prod_{j < i} X_j & \xleftarrow{\sigma_{\leq i}} & \prod_{j \leq i} X_j & \xlongequal{\quad} & \prod_{j \leq i} X_j \\ \downarrow & & \downarrow & & \downarrow \\ \prod_{j \in \text{pa}(i)} X_j & \xleftarrow{\sigma} & \prod_{j \in \text{pa}(i)} X_j \times X_i & \longrightarrow & X_i \end{array}.$$

The upper decorated span is the horizontal morphism corresponding to Fong’s factor. Finally, note that whereas traditional statistical modelling is restricted to sections of product types (as above), using $\mathcal{K}$ and $\mathbb{S}$ means we can build probabilistic models involving general dependent types.

3 Undirected models: open factor graphs

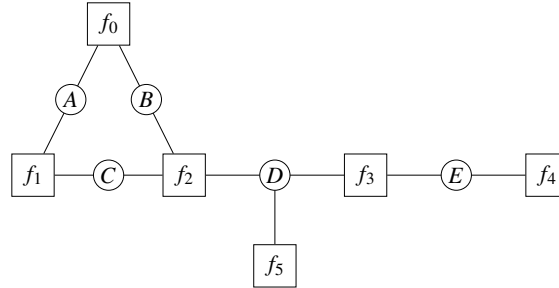


Figure 1: A factor graph, in the style of Wainwright and Jordan [23, §2.1.3].

The preceding section explained how copy-composition is central to Bayesian networks: *directed* probabilistic graphical models. In this section, we shall see that it is also central to the construction of undirected models known as *factor graphs* [6, 23]. A factor graph is a graphical representation of the factorization of a function, which is often but not necessarily interpreted as the density of a probability distribution. Thus Figure 1 represents a function f of the form

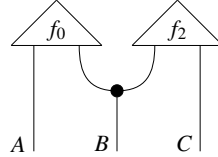
$$f(a, b, c, d, e) = f_0(a, b) f_1(a, c) f_2(b, c, d) f_3(d, e) f_4(e) f_5(d)$$

where the variables have the types $a : A, b : B, c : C, d : D$. Note that we could see f as composed of ‘factors’ f_i ; it is this composition that we explore here.

Typically, the factors of a factor graph are functions into some object of scalars; for example, we might interpret f_2 (above) to have the type $B \times C \times D \rightarrow \mathbb{R}_+$, for some spaces B, C, D . Should these spaces be $\mathbb{R}_+$ -modules, then we might suppose f_2 to be a linear functional or *covector* on their product.

$\mathbb{R}_+$ -modules form a category $\mathbf{Mod}_{\mathbb{R}_+}$ which is equipped with a tensor product $\otimes$, whose unit is the 1-dimensional space of scalars $\mathbb{R}_+$, often denoted I . Covectors on X then correspond to *costates*, morphisms $X \rightarrow I$. This suggests that we may internalize the notion of factor in a monoidal category $(\mathcal{C}, \otimes, I)$ simply as the costates in $\mathcal{C}$.

This first step towards an abstract treatment should be taken with care, as any semicartesian monoidal category, such as **Set** or any Markov category (in the sense of Fritz [10]), will only have trivial costates, the monoidal unit there being by definition terminal. The composition of factors then proceeds by copy-composition; notice for example the duplication of the variable $b : B$ in the expression for f above, or in the following string-diagrammatic representation of its f_0, f_2 factor:



This suggests that an appropriate categorical setting for factor graphs is that of *copy-discard categories*⁴, of which **sfKrn** (or any fibre of $\mathcal{K}$) is an example.

Because factor graphs are a form of undirected graphical model, it is natural to expect them to fit into the framework of undirected wiring diagrams [20] [24, Chapter 7]. Undirected wiring diagrams are algebras for an operad of cospans [9], and one way to obtain such algebras is through the decorated cospans construction, originally due to Fong [8]. (But note that the decorated spans of the previous section are not undirected in the same way, because the decorations there are asymmetric.) The main result of this section is to show that ‘open’ factor graphs may be obtained and composed precisely through the double-categorical generalization of this construction due to Patterson [18].

Remark 3.1. Fong and Spivak [9] show that cospan-algebras are equivalent to hypergraph categories, in which each morphism has the form of a hypergraph. Consequently, this is also the case for factor graphs.

The remainder of this section is dedicated to constructing a double category indexed by cospans, *i.e.* a double pseudo functor F , and instantiating its double Grothendieck construction, the double category $\mathbb{F}G$. The domain of F is the double category of cospans of finite sets, whose vertical category $\mathbf{Cospans}(\mathbf{FinSet})_0$ is simply **FinSet**, and whose horizontal category $\mathbf{Cospans}(\mathbf{FinSet})_1$ is the category of diagrams in **FinSet** of shape $\bullet \rightarrow \bullet \leftarrow \bullet$, *i.e.* the functor category $\mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}}$. The codomain of F is $\mathbf{Span}(\mathbf{Cat})$, whose vertical 2-category is **Cat**, and whose horizontal 2-category is $\mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}}$.

Henceforth, we will assume an ambient copy-discard category $(\mathcal{C}, I, \otimes)$. Because finite sets are not ordered and we plan to define factors over accordingly indexed monoidal products, we must additionally assume that the monoidal structure $(I, \otimes)$ on $\mathcal{C}$ is symmetric.

There will be fewer simplifications involved in the definition of F (versus S), and so we split the construction into four subsections.

3.1 Vertical decoration: factors’ interfaces

The pseudo functor $F_0 : \mathbf{FinSet} \rightarrow \mathbf{Cat}$ assigns categories of decorations to finite sets. For us, these decorations will be the exposed interfaces of factors, given by a discrete diagram in $\mathcal{C}$: a functor $\text{disc } X \rightarrow$

⁴A *copy-discard category* is a monoidal category in which every object is equipped with a comonoid structure (with respect to the ambient monoidal structure) [2]. A *Markov category* is a semicartesian copy-discard category [10]. Precisely because of the existence of non-trivial costates, copy-discard categories often end up being more useful in modelling practice than Markov categories, despite the utility of the latter for synthetic probability theory.

$\mathcal{C}$, for some finite set X , where $\text{disc}X$ is the discrete category on X . However, were we just to define F_0X to be the functor category $\mathbf{Cat}(\text{disc}X, \mathcal{C})$, we would run into problems later with the composition of 2-cells. This is because vertical morphisms — decorated by morphisms in the image of F_0 — may be used to transform the interface types, but the composition of factors proceeds by copying. For this to be compatible with morphisms of interfaces, we must restrict the latter to be comonoid homomorphisms, which (in probabilistic contexts) we can often understand as deterministic functions.

Therefore, on objects $X : \mathbf{FinSet}$, let F_0X be the functor category $\mathbf{Cat}(\text{disc}X, \mathbf{Comon}(\mathcal{C}, I, \otimes))$ where $\mathbf{Comon}(\mathcal{C}, I, \otimes)$ denotes the (wide) monoidal subcategory of $\mathcal{C}$ whose morphisms are the comonoid homomorphisms. Note that, because the objects χ of F_0X are functors on a discrete category, a morphism $\varphi : \chi \rightarrow \chi'$ (hence a natural transformation) is simply given by an X -indexed family of comonoid homomorphisms $\varphi_x : \chi(x) \rightarrow \chi'(x)$ in $\mathcal{C}$.

Given a morphism $f : X \rightarrow Y$ in $\mathbf{FinSet}$, we let F_0 return a functor $F_0X \rightarrow F_0Y$ which we denote by f_* and define as follows. If χ is an object of F_0X then we define $f_*\chi$ by the mapping $y \mapsto \otimes_{x:f^{-1}(y)} \chi(x)$ (for $y : Y$). Then, if $\varphi : \chi \rightarrow \chi'$ is a morphism of F_0X , we define each y -component $(f_*\varphi)_y$ of $f_*\varphi$ to be $\otimes_{x:f^{-1}(y)} \varphi_x$. Note that this definition yields components of the correct type

$$(f_*\varphi)_y : (f_*\chi)(y) = \otimes_{x:f^{-1}(y)} \chi(x) \xrightarrow{\otimes_{x:f^{-1}(y)} \varphi_x} \otimes_{x:f^{-1}(y)} \chi'(x) = (f_*\chi')(y).$$

Naturality is trivially satisfied, because χ and χ' are discrete functors, and the functoriality of the reindexing f_* follows from the functoriality of $\otimes$. Likewise, the pseudofunctoriality of F_0 itself follows from the functoriality of inverse images and the functoriality and symmetry of $\otimes$, so that $\otimes_{y:g^{-1}(z)} \otimes_{x:f^{-1}(y)} \cong \otimes_{x:(g \circ f)^{-1}(z)}$.

3.2 Horizontal decoration: ‘open’ factors

The pseudo functor $F_1 : \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} \rightarrow \mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}}$ assigns spans of categories to cospans of finite sets, compatibly with F_0 . We interpret a cospan $A \xrightarrow{a} X \xleftarrow{b} B$ as follows. The apex set X will be decorated with two kinds of data: first, a category of interfaces suited to factors in $\mathcal{C}$; and second, for each suitable interface, a category of factors on that interface (compatibly with morphisms of interfaces). The legs of the cospan a and b will be used to expose those parts of the interface which are open to composition, and in this way the objects A and B will be decorated by F_0A and F_0B respectively; this is one of the indexed-double-categorical coherence axioms (‘well-definition’).

Thus, given a cospan $m = (A \xrightarrow{a} X \xleftarrow{b} B)$, we let F_1 return a span of categories $F_0A \xleftarrow{\pi_a} \int F_1 m \xrightarrow{\pi_b} F_0B$. The apex category $\int F_1 m$ has for objects pairs (χ, f) where $\chi : \text{disc}X \rightarrow \mathcal{C}$ is an interface and $f : \chi^\otimes \rightarrow I$ is a factor on that interface. $\chi^\otimes$ denotes the monoidal product $\otimes_{x:X} \chi(x)$ in $\mathcal{C}$. The morphisms of $\int F_1 m$ are factorizations of factors through transformations of interfaces; that is to say, a morphism $(\chi, f) \rightarrow (\chi', f')$ in $\int F_1 m$ is a morphism of interfaces $\varphi : \chi \rightarrow \chi'$ such that $f = \varphi^* f'$, where φ^* denotes precomposition by $\varphi^\otimes := \otimes_{x:X} \varphi_x$. The functors π_a and π_b forget the factors and project out those parts of the interfaces over X that are exposed by a and b respectively. ($\int F_1 m$ and the functors π_a, π_b are in fact obtained through universal constructions — a limit and a Grothendieck construction — as we explain in Appendix B.1. Universality then yields the functorial action of F_1 on morphisms of cospans.)

3.3 Factor composition

The functors F_0 and F_1 describe how to attach factors to interfaces, and how these interfaces (and the factors on them) may be transformed by morphisms in $\mathcal{C}$. But they do not tell us about the operation

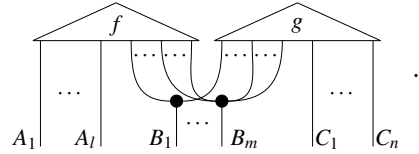
at the heart of the open factor graph construction: the horizontal (copy-)composition of factors, or the horizontal composition of their morphisms. These operations are formalized by the external composition structure (μ, η) with which we now equip F .

Let us begin with μ , a pseudo natural transformation whose component $\mu_{m,n}$ at a pair of composable cospans $m = (A \xrightarrow{a} X \xleftarrow{b} B)$ and $n = (B \xrightarrow{b'} Y \xleftarrow{c} C)$ must be a functor $\int F_1 m \times_{F_0 B} \int F_1 n \rightarrow \int F_1 (n \diamond m)$. Given factors $(\chi, f), (\gamma, g)$ in its domain, $\mu_{m,n}$ must return an interface over $X +_B Y$ and a factor on that interface. The former is given by the copairing $[\chi, \gamma]_B$ of χ and γ over B , while the latter is given by copy-composition. For each $j : X +_B Y$, copying induces a morphism

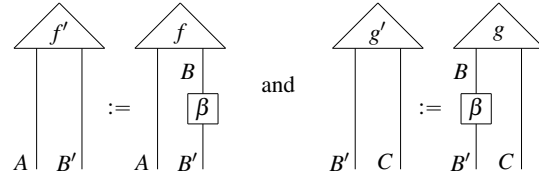
$$\delta_{\chi, \gamma}^j : [\chi, \gamma]_B(j) \rightarrow \bigotimes_{i: [t_B^X, t_B^Y]^{-1}(j)} [\chi, \gamma](i)$$

in $\mathcal{C}$ which is unique up to coassociativity. We then define $\delta_{\chi, \gamma} := \bigotimes_{j: X +_B Y} \delta_{\chi, \gamma}^j$, which has the type $[\chi, \gamma]_B^{\otimes} \rightarrow [\chi, \gamma]^{\otimes}$. We can compose $f \otimes g$ after this higher-arity copier to yield a factor on the interface $[\chi, \gamma]_B$. Thus $\mu_{m,n}((\chi, f), (\gamma, g)) := ([\chi, \gamma]_B, (f \otimes g) \circ \delta_{\chi, \gamma})$. (For a more detailed construction and proof that this yields a pseudo natural transformation, see Appendix B.2.)

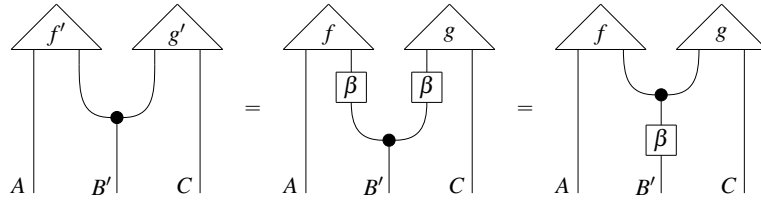
To see μ in action, suppose A has l -many elements, C has n -many elements, and the coupled interface B has m -many elements, with B_1 shared once between f and g but B_m shared an arbitrary number of times.



The naturality of $\mu_{m,n}$ means that if we were to have f' and g' defined by transforming the interface B along β like so



then copy-composing f' and g' along B' yields the same as copy-composing f and g along B and then transforming the B interface of the result accordingly:



However, this only works for ‘deterministic’ transformations β : although the rules of open factor graphs do allow us to transform factors using arbitrary morphisms, we can only pull these transformations ‘outside’ the composites when the transformations are comonoid homomorphisms.

It remains to define the unit η , a pseudo natural transformation whose component η_X at each finite set X is given by a functor $\eta_X : F_0 X \rightarrow \int F_1 (\text{id}_X)$. We define η_X using the counits of the comonoid structures on the interfaces on X . That is to say, η_X maps an interface $\chi : \text{disc} X \rightarrow \mathbf{Comon}(\mathcal{C})$ to the pair (χ, ϵ_χ) ,

where here ε_χ denotes the canonical discarding map $\chi^\otimes \rightarrow I$. The functorial action of η_X is obtained from the fact that all morphisms in F_0X are comonoid homomorphisms, so if φ is a morphism $\chi \rightarrow \chi'$ then $\varepsilon_\chi = \varphi^* \varepsilon_{\chi'}$. Pseudo naturality of η obtains likewise, and its unitality with respect to μ follows from the counitality of the comonoid counits.

3.4 Putting it all together

We have defined a double pseudo functor $F : \mathbf{Cospans}(\mathbf{FinSet}) \rightarrow \mathbf{Span}(\mathbf{Cat})$, from which we obtain a double category of open factor graphs $\mathbb{FG}$, doubly opfibred over $\mathbf{Cospans}(\mathbf{FinSet})$, by applying the double Grothendieck construction. This gives $\mathbb{FG}$ the following structure:

1. Its objects (0-cells) are pairs (A, α) where A is a finite set and α is an object of F_0A — *i.e.*, an interface in $\mathcal{C}$ on A .
2. A vertical 1-cell $(A, \alpha) \rightarrow (A', \alpha')$ is a pair (f, φ) of a map $f : A \rightarrow A'$ and a deterministic transformation of interfaces $\varphi : f_*\alpha \rightarrow \alpha'$ — where $f_* = F_0(f)$ so that φ is a morphism in F_0A' .
3. A horizontal 1-cell $(A, \alpha) \rightarrow (B, \beta)$ is a quintuple (X, a, b, χ, p) where $(A \xrightarrow{a} X \xleftarrow{b} B)$ is a cospan of finite sets, χ is an interface on X , and p is a factor on χ — *i.e.*, (χ, p) is an object of $\int \underline{F}_1(X, a, b)$ — such that $\pi_a(\chi, p) = \alpha$ and $\pi_b(\chi, p) = \beta$.

4. A 2-cell as in the diagram

$$\begin{array}{ccc}
 (A, \alpha) & \xrightarrow{(X, a, b, \chi, p)} & (B, \beta) \\
 (f^l, \varphi^l) \downarrow & (f, \varphi) & \downarrow (f^r, \varphi^r) \\
 (A', \alpha') & \xrightarrow{(X', a', b', \chi', p')} & (B', \beta')
 \end{array}$$
 is a pair (f, φ) where $f : X \rightarrow X'$

is a map of finite sets and φ is a transformation of interfaces $f_*\chi \rightarrow \chi'$, such that $f_*p = \varphi^*p'$, $\pi_{a'}(\varphi) = \varphi^l$, $\pi_{b'}(\varphi) = \varphi^r$ (with φ^l and φ^r deterministic), and such that (f, f^l, f^r) constitutes a morphism of cospans, meaning that $a' \circ f^l = f \circ a$ and $b' \circ f^r = f \circ b$.

5. Vertical 1-cells $(f, \varphi) : (A, \alpha) \rightarrow (A', \alpha')$ and $(f', \varphi') : (A', \alpha') \rightarrow (A'', \alpha'')$ compose to yield $(f' \circ f, \varphi' \circ \varphi^l \varphi) : (A, \alpha) \rightarrow (A'', \alpha'')$; the vertical composition of 2-cells is similar. The vertical identity 1-cell on (A, α) is the pair $(\text{id}_A, \text{id}_\alpha)$ of identity morphisms $\text{id}_A : A \rightarrow A$ and $\text{id}_\alpha : \alpha \rightarrow \alpha$. We denote vertical composition by $\circ$.
6. Finally, horizontal 1-cells (and 2-cells, horizontally) compose by cospan- and copy-composition, as described in §3.3. Horizontal identities are given by the external unit η . We denote horizontal composition by $\diamond$.

All told, this (plus the details in Appendix B) yields the following theorem.

Theorem 3.2 (Open factor graphs). Given a symmetric monoidal copy discard category $(\mathcal{C}, I, \otimes)$, the foregoing data define a pseudo double category $\mathbb{FG}$ doubly opfibred over $\mathbf{Cospans}(\mathbf{FinSet})$.

Remark 3.3. Since each of the elements of the construction of $\mathbb{FG}$ are functorial, we expect the construction itself to be functorial, so that a functor $\mathcal{C} \rightarrow \mathcal{C}'$ yields a double functor $\mathbb{FG}_\mathcal{C} \rightarrow \mathbb{FG}_{\mathcal{C}'}$. But we have not established this, and leave it to future work.

In other future work, we hope to make use of $\mathbb{FG}$ to study the compositionality of inference on factor graphs, and to study the formal relationships between directed and undirected models.

3.5 On a graphical calculus

We motivated the construction of $\mathbb{FG}$ with the graphic Figure 1. Decorated cospan categories as originally formulated inherit a graphical calculus from their status as hypergraph categories or undirected wiring diagram algebras. Lacking a formal theory of double operads and their graphical calculi, we do not have a similarly formal graphical calculus for $\mathbb{FG}$; still, we expect that its horizontal bicategory (appropriately quotiented) would fit the undirected wiring diagram pattern, at the cost of the transformations of interfaces. Inspired by this, and in an attempt not to pay that cost, in Appendix B.3 we (informally) propose a graphical calculus that matches and extends the non-compositional depictions in the statistics literature.

4 References

- [1] Matteo Capucci, Bruno Gavranović, Jules Hedges, and Eigil Fjeldgren Rischel. “Towards Foundations of Categorical Cybernetics”. In: *Electronic Proceedings in Theoretical Computer Science*. Applied Category Theory 2021. Vol. 372. Cambridge, UK: Open Publishing Association, 11/03/2022, pp. 235–248. DOI: 10.4204/EPTCS.372.17. URL: <http://arxiv.org/abs/2105.06332v2> (visited on 09/29/2023).
- [2] Kenta Cho and Bart Jacobs. “Disintegration and Bayesian Inversion via String Diagrams”. In: *Math. Struct. Comp. Sci.* 29 (2019) 938–971 (08/29/2017). DOI: 10.1017/S0960129518000488. arXiv: 1709.00322v3 [cs.AI].
- [3] Kenta Cho, Bart Jacobs, Bas Westerbaan, and Abraham Westerbaan. “An Introduction to Effectus Theory”. 2015. arXiv: 1512.05813 [cs.LO].
- [4] Bob Coecke and Aleks Kissinger. “Categorical Quantum Mechanics II: Classical-Quantum Interaction”. 05/27/2016. arXiv: 1605.08617v1 [quant-ph].
- [5] Geoffrey Cruttwell, Michael Lambert, Dorette Pronk, and Martin Szyld. *Double Fibrations*. 05/30/2022. arXiv: 2205.15240 [math]. URL: <http://arxiv.org/abs/2205.15240> (visited on 03/20/2023). preprint.
- [6] Justin Dauwels. “On Variational Message Passing on Factor Graphs”. In: *2007 IEEE International Symposium on Information Theory*. 2007 IEEE International Symposium on Information Theory. Nice: IEEE, 06/2007, pp. 2546–2550. ISBN: 978-1-4244-1397-3. DOI: 10.1109/ISIT.2007.4557602. URL: <http://ieeexplore.ieee.org/document/4557602/> (visited on 11/14/2023).
- [7] Brendan Fong. “Causal Theories: A Categorical Perspective on Bayesian Networks”. University of Oxford, 2013-01-26, 2013. arXiv: 1301.6201v1 [math.PR]. URL: <http://arxiv.org/abs/1301.6201>.
- [8] Brendan Fong. “Decorated Cospans”. In: *Theory and Applications of Categories* 30.33 (2015), pp. 1096–1120. arXiv: 1502.00872. URL: <http://www.tac.mta.ca/tac/volumes/30/33/30-33abs.html>.

- [9] Brendan Fong and David I. Spivak. “Hypergraph Categories”. In: *Journal of Pure and Applied Algebra* 223.11 (11/2019), pp. 4746–4777. ISSN: 00224049. DOI: 10.1016/j.jpaa.2019.02.014. arXiv: 1806.08304v3 [math.CT]. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0022404919300489> (visited on 12/17/2022).
- [10] Tobias Fritz. “A Synthetic Approach to Markov Kernels, Conditional Independence and Theorems on Sufficient Statistics”. In: *Advances in Mathematics* 370.107239 (08/19/2019). DOI: 10.1016/j.aim.2020.107239. arXiv: 1908.07021v3 [math.ST].
- [11] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. “A Convenient Category for Higher-Order Probability Theory”. In: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)* (06/2017). DOI: 10.1109/lics.2017.8005137. arXiv: 1701.02547 [cs.PL].
- [12] Bart Jacobs. *Categorical Logic and Type Theory*. Vol. 141. Studies in Logic and the Foundations of Mathematics. North-Holland Publishing Co., Amsterdam, 1999, pp. xviii+760. ISBN: 0-444-50170-3.
- [13] Olav Kallenberg. *Foundations of Modern Probability*. 2nd ed. Probability and Its Applications. New York: Springer, 2002. 638 pp. ISBN: 978-0-387-94957-4.
- [14] Alexander S. Kechris. *Classical Descriptive Set Theory*. Vol. 156. Graduate Texts in Mathematics. New York, NY: Springer New York, 1995. ISBN: 978-1-4612-8692-9 978-1-4612-4190-4. DOI: 10.1007/978-1-4612-4190-4. URL: <http://link.springer.com/10.1007/978-1-4612-4190-4> (visited on 10/11/2024).
- [15] Magnus Koudahl, Thijs van de Laar, and Bert de Vries. *Realising Synthetic Active Inference Agents, Part I: Epistemic Objectives and Graphical Specification Language*. 06/13/2023. arXiv: 2306.08014 [cs]. URL: <http://arxiv.org/abs/2306.08014> (visited on 07/28/2023). preprint.
- [16] Joe Moeller and Christina Vasilakopoulou. “Monoidal Grothendieck Construction”. In: *Theory and Applications of Categories* 35.31 (2020), pp. 1159–1207. arXiv: 1809.00727v2 [math.CT].
- [17] nLab authors. *Categorical Semantics of Dependent Type Theory*. 06/2024. URL: <https://ncatlab.org/nlab/show/categorical+semantics+of+dependent+type+theory>.
- [18] Evan Patterson. *Structured and Decorated Cospans from the Viewpoint of Double Category Theory*. 04/02/2023. arXiv: 2304.00447 [math]. URL: <http://arxiv.org/abs/2304.00447> (visited on 04/20/2023). preprint.
- [19] Urs Schreiber. “Quantization via Linear Homotopy Types”. 02/27/2014. arXiv: 1402.7041 [math-ph].

- [20] David I. Spivak. *The Operad of Wiring Diagrams: Formalizing a Graphical Language for Databases, Recursion, and Plug-and-Play Circuits*. 05/01/2013. arXiv: 1305.0297 [cs, math]. URL: <http://arxiv.org/abs/1305.0297> (visited on 12/23/2022). preprint.
- [21] Toby St Clere Smithe. “Approximate Inference via Fibrations of Statistical Games”. In: *Electronic Proceedings in Theoretical Computer Science*. Applied Category Theory 2023. Vol. 397. College Park, MD: Open Publishing Association, 06/29/2023, pp. 279–298. DOI: 10.4204/EPTCS.397.17. arXiv: 2306.17009 [math, stat]. URL: <http://arxiv.org/abs/2306.17009> (visited on 09/11/2023).
- [22] Matthijs Vákár and Luke Ong. *On S-Finite Measures and Kernels*. 10/03/2018. arXiv: 1810.01837 [math.PR]. Pre-published.
- [23] Martin J. Wainwright and Michael I. Jordan. “Graphical Models, Exponential Families, and Variational Inference”. In: *Foundations and Trends® in Machine Learning* 1.1–2 (2007), pp. 1–305. ISSN: 1935-8245. DOI: 10.1561/22000000001. URL: <http://dx.doi.org/10.1561/22000000001>.
- [24] Donald Yau. *Operads of Wiring Diagrams*. Vol. 2192. Lecture Notes in Mathematics. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-95000-6 978-3-319-95001-3. DOI: 10.1007/978-3-319-95001-3. arXiv: 1512.01602 [math]. URL: <http://link.springer.com/10.1007/978-3-319-95001-3> (visited on 09/11/2023).

A Directed models

A.1 Kernels bifibration

Proof of Proposition 2.7. By randomization [13, Lemma 2.22], there is a measurable function $\rho : \mathbb{I} \rightarrow \mathbb{I} \times \mathbb{I}$ such that $\rho_* v = v \otimes v$. Suppose h, k have the types $(E, p) \rightsquigarrow^h (F, q) \rightsquigarrow^k (G, r)$, and suppose $\bar{h} : E \rightarrow M_F$ and $\bar{k} : F \rightarrow M_G$ represent them as indexed random variables, such that $\bar{h}(x)_* v = h(x)$ and $\bar{k}(y)_* v = k(y)$. The composite kernel $k \circ h$ may then be represented by the indexed random variable $kh : E \rightarrow M_G$ defined by $x \mapsto r \mapsto k(h(x)(\rho(r)_1))(\rho(r)_2)$ where $(-)_1$ and $(-)_2$ represent the first and second projections out of the product; that is, we have $(k \circ h)(x) = kh(x)_* v$.

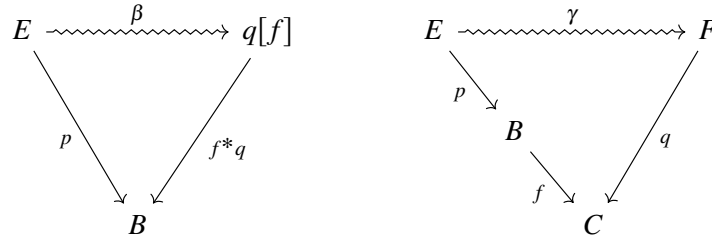
From Proposition 2.6, we know that if $x \in p[b]$, then $h(x)(s) \in q[b]$ for all $s \in \mathbb{I}$. Similarly, if $y \in q[b]$, then $k(y)(s) \in r[b]$ for all $s \in \mathbb{I}$. So, setting $y = h(x)(\rho(r)_1)$, we must have $k(h(x)(\rho(r)_1))(s) \in r[b]$ for all $s \in \mathbb{I}$, and in particular for $s = \rho(r)_2$. Thus $kh(x)(s) \in r[b]$, for all $s \in \mathbb{I}$, as required. $\square$

Proof of Proposition 2.11. We need to exhibit an isomorphism

$$\begin{array}{ccc} & \xrightarrow{(-)^b} & \\ \mathcal{K}_B(p, \Delta_f q) & & \mathcal{K}_C(\Sigma_f p, q) \\ & \xleftarrow{(-)^\#} & \end{array}$$

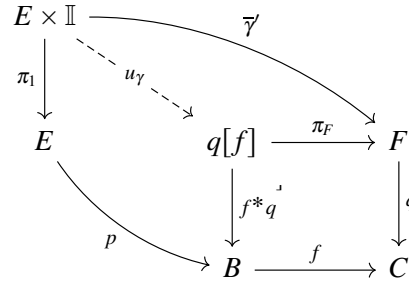
natural in p and q .

Suppose then that we have $\beta : p \rightsquigarrow \Delta_f q$ in $\mathcal{K}_B$ and $\gamma : \Sigma_f p \rightsquigarrow q$ in $\mathcal{K}_C$ as in the following diagrams:



Let $\bar{\beta} : E \rightarrow \mathbb{I} \rightarrow q[f]$ represent β as an indexed random variable, and let π_F be the projection $q[f] \rightarrow F$. Then we may define β^b as represented by the map $\bar{\beta}^b : E \rightarrow \mathbb{I} \rightarrow F$ defined by $\bar{\beta}^b(x)(r) = \pi_F(\bar{\beta}(x)(r))$.

Dually, define $\gamma^\sharp : p \rightsquigarrow \Delta_f q$ by the representative $\bar{\gamma}^\sharp : E \rightarrow \mathbb{I} \rightarrow q[f] : x \mapsto r \mapsto u_\gamma(x, r)$ where u_γ is obtained from the universal property of the pullback in the following diagram (as the dashed arrow)



where π_1 is the projection out of the product and $\bar{\gamma} : E \times \mathbb{I} \rightarrow F$ is the uncurrying of the representative $\bar{\gamma} : E \rightarrow \mathbb{I} \rightarrow F$ of γ .

From these definitions, it is clear that β^b and $\gamma^\sharp$ are well-defined morphisms in $\mathcal{K}$, in the sense that the relevant triangles commute, so it remains to check the natural isomorphism. That the definitions yield an isomorphism of hom sets is similarly immediate. (On the one hand, let $\gamma = \beta^b$ in the diagram for u_γ above, and observe that $\gamma^\sharp$ must equal β . On the other, let $\beta = \gamma^\sharp$ and observe that by construction β^b must equal γ .)

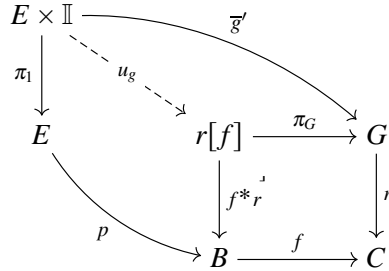
It therefore only remains to demonstrate naturality. Since $(-)^b$ and $(-)^^\sharp$ are mutually inverse, we need only demonstrate naturality for one; we choose $(-)^^\sharp$. This means showing that the following square commutes for all kernels φ, ψ of appropriate type:

$$\begin{array}{ccc}
 \mathcal{K}_C(\Sigma_f p, q) & \xrightarrow{(-)_{p,q}^\sharp} & \mathcal{K}_B(p, \Delta_f q) \\
 \downarrow \mathcal{K}_C(\Sigma_f \varphi, \psi) & & \downarrow \mathcal{K}_B(\varphi, \Delta_f \psi) \\
 \mathcal{K}_C(\Sigma_f o, r) & \xrightarrow{(-)_{o,r}^\sharp} & \mathcal{K}_B(o, \Delta_f r)
 \end{array}$$

Since the hom functors $\mathcal{K}_C$ and $\mathcal{K}_B$ act by pre- and post-composition, this requires establishing that

$$o \rightsquigarrow \varphi p \rightsquigarrow \gamma^\sharp \Delta_f q \rightsquigarrow \Delta_f \psi \Delta_f r = \left(\Sigma_f o \rightsquigarrow \Sigma_f \varphi \Sigma_f p \rightsquigarrow \gamma q \rightsquigarrow \psi r \right)^\sharp.$$

Let the composite on the right-hand side be denoted by g . Randomization gives us a function $\rho : \mathbb{I} \rightarrow \mathbb{I} \times \mathbb{I} \times \mathbb{I}$ such that $\rho_* v = v \otimes v \otimes v$. Let $\rho(s) = (s_1, s_2, s_3)$. A representative $\bar{g}$ of g is given by $\bar{g}(x)(s) = \bar{\psi}(\bar{\gamma}(\bar{\phi}(x)(s_1))(s_2))(s_3)$ where $\bar{\psi}$, $\bar{\gamma}$, and $\bar{\phi}$ are representatives of ψ , γ , and ϕ respectively. $g^\sharp$ is then represented by the currying of the dashed map u_g in the following diagram:



Let the composite on the left-hand side (above) be denoted by h . A representative $\bar{h}$ of h is given by

$$\begin{aligned} \bar{h}(x)(s) &= \bar{\psi}[f](\bar{\gamma}^\sharp(\bar{\phi}(x)(s_1))(s_2))(s_3) \\ &= \bar{\psi}[f](u_\gamma(\bar{\phi}(x)(s_1), s_2))(s_3) \\ &= \bar{\psi}(\pi_F(u_\gamma(\bar{\phi}(x)(s_1), s_2)))(s_3) \end{aligned}$$

so we need to verify that $\pi_F(u_\gamma(\bar{\phi}(x)(s_1), s_2)) = \bar{\gamma}(\bar{\phi}(x)(s_1))(s_2)$. We have $\pi_F(u_\gamma(y, t)) = \bar{\gamma}(y)(t)$ by construction, so $\bar{h}(x)(s) = \bar{g}^\sharp(x)(s)$. This establishes that $\Sigma_f \dashv \Delta_f$. $\square$

Proof of Proposition 2.12. The functor $\Sigma_\rho \Delta_\pi : \mathcal{K}_E \rightarrow \mathcal{K}_F$ sends $X \xrightarrow{\alpha} E$ to $\pi^* X \xrightarrow{\Delta_\pi \alpha} P \xrightarrow{\rho} F$; fibrewise, by adjointness, we have $\Sigma_\rho \Delta_\pi(\alpha)[i] \cong \alpha[\pi \circ \rho^* i]$ and, on morphisms f , we have $\Sigma_\rho \Delta_\pi(f)[i] = f[\pi \circ \rho^* i]$. On the other side, the functor $\Delta_q \Sigma_p$ sends α to $q^* X \xrightarrow{\Delta_q \alpha} q^* E \xrightarrow{\Delta_q p} F$; fibrewise, we have $\Delta_q \Sigma_p(\alpha)[i] \cong \alpha[p^*(q \circ i)]$ on objects and $\Delta_q \Sigma_p(f)[i] = f[p^*(q \circ i)]$ on morphisms. It therefore suffices to show that $p^*(q \circ i) = \pi \circ \rho^* i$ in **QBS**. This follows from the universal property of the pullback, which implies that the pasting of two pullback squares is again a pullback square:

$$\begin{array}{ccccc} & & p^*(q \circ i) & & \\ & \nearrow & & \searrow & \\ \bullet & \xrightarrow{\rho^* i} & P & \xrightarrow{\pi} & E \\ \downarrow \scriptstyle \lrcorner & & \downarrow \scriptstyle \lrcorner \rho & & \downarrow \scriptstyle \lrcorner p \\ I & \xrightarrow{i} & F & \xrightarrow{q} & B \end{array}$$

The outer and inner squares are thus both pullbacks of p along $q \circ i$, and hence we must have $p^*(q \circ i) = \pi \circ \rho^* i$. This yields an isomorphism $\Sigma_\rho \Delta_\pi(\alpha) \cong \Delta_q \Sigma_p(\alpha)$ rather than an equality because universal constructions are only unique up to natural isomorphism; and thus the naturality of this isomorphism follows similarly from universality. $\square$

A.2 Pull-push sections

Proof of Theorem 2.15. Much of the structure is simplified, owing to the trivial vertical decorations and the decoration of the apices by sets. Thus, for example, each component $\mu_{m,n}$ or η_E of the pseudo natural transformations μ or η is only a function (discrete functor). In the main text, the functoriality of S_0

and S_1 is justified. So here we will only justify the (pseudo) naturality of μ and η and their (pseudo) associativity and unitality.

Suppose then that $f : m \rightarrow m'$ and $g : n \rightarrow n'$ are horizontally composable morphisms of spans (*i.e.*, such that $g_l = f_r$) from $m = (A \xleftarrow{a} E \xrightarrow{b} B)$ and $n = (B \xleftarrow{\bar{b}} F \xrightarrow{c} C)$ to $m' = (A' \xleftarrow{a'} E' \xrightarrow{b'} B')$ and $n' = (B' \xleftarrow{\bar{b}'} F' \xrightarrow{c'} C')$. To validate the pseudo naturality of μ is to confirm the commutativity of the square

$$\begin{array}{ccc} S_1(m') \times S_1(n') & \xrightarrow{\mu_{m',n'}} & S_1(n' \diamond m') \\ S_1(f) \times S_1(g) \downarrow & & \downarrow S_1(g \diamond f) \\ S_1(m) \times S_1(n) & \xrightarrow{\mu_{m,n}} & S_1(n \diamond m) \end{array} \cdot$$

up to natural isomorphism. The top-right composite evaluates to

$$\begin{aligned} S_1(g \diamond f)(\Sigma_{a'} \Delta_{b'}(\tau') \circ \sigma') &= \Delta_{f_l}(\Sigma_{a'} \Delta_{b'}(\tau') \circ \sigma') && \text{(by definition)} \\ &= \Delta_{f_l} \Sigma_{a'} \Delta_{b'}(\tau') \circ \Delta_{f_l}(\sigma') && \text{(by functoriality)} \end{aligned}$$

while the left-bottom composite evaluates to

$$\begin{aligned} \Sigma_a \Delta_b(S_1(g)(\tau')) \circ S_1(f)(\sigma') &= \Sigma_a \Delta_b \Delta_{g_l}(\tau') \circ \Delta_{f_l}(\sigma') && \text{(by definition)} \\ &= \Sigma_a \Delta_f \Delta_{b'}(\tau') \circ \Delta_{f_l}(\sigma') && \text{(since } g_l \circ b = b' \circ f) \\ &\cong \Delta_{f_l} \Sigma_{a'} \Delta_{b'}(\tau') \circ \Delta_{f_l}(\sigma') && \text{(by Beck-Chevalley)} \end{aligned}$$

and so $S_1(g \diamond f) \circ \mu_{m',n'} \cong \mu_{m,n} \circ (S_1(f) \times S_1(g))$ as required.

Validating the pseudo naturality of η means, for any $f : E \rightarrow F$ in $\mathcal{B}$, validating that $\eta_F \cong S_1(f) \circ \eta_E$, *i.e.*, $\eta_F \cong \Delta_f \eta_E$. Since η_E is the identity on ιE and Δ_f is a functor, it must preserve identities. So in fact η is strictly natural (just as pull-push copy-composition is strictly unital, as we confirm below).

To validate pseudo associativity, suppose m and n are spans as before and $o = (C \xleftarrow{\bar{c}} G \xrightarrow{d} D)$. We need to show that $\mu_{n \diamond m, o}(\mu_{m,n}(\sigma, \tau), \rho) \cong \mu_{m, o \diamond n}(\sigma, \mu_{n,o}(\tau, \rho))$ for all appropriately typed sections σ, τ, ρ . This obtains as follows:

$$\begin{aligned} \mu_{n \diamond m, o}(\mu_{m,n}(\sigma, \tau), \rho) &= \mu_{n \diamond m, o}(\Sigma_a \Delta_b(\tau) \circ \sigma, \rho) && \text{(by definition)} \\ &= \Sigma_a \Sigma_{\pi_E} \Delta_{\pi_F} \Delta_c(\rho) \circ \Sigma_a \Delta_b(\tau) \circ \sigma && \text{(by definition)} \\ &\cong \Sigma_a \Delta_b \Sigma_{\bar{b}} \Delta_c(\rho) \circ \Sigma_a \Delta_b(\tau) \circ \sigma && \text{(by Beck-Chevalley)} \\ &= \Sigma_a \Delta_b(\Sigma_{\bar{b}} \Delta_c(\rho) \circ \tau) \circ \sigma && \text{(by functoriality)} \\ &= \mu_{m, o \diamond n}(\sigma, \Sigma_{\bar{b}} \Delta_c(\rho) \circ \tau) && \text{(by definition)} \\ &= \mu_{m, o \diamond n}(\sigma, \mu_{n,o}(\tau, \rho)) && \text{(by definition)} \end{aligned}$$

where π_E and π_F are the projections out of the pullback of $\bar{b}$ along b .

Finally, strict unitality demands that $\mu_{\text{id}_B, n}(\eta_B, \tau) = \tau$ and $\mu_{m, \text{id}_B}(\sigma, \eta_B) = \sigma$. We have

$$\mu_{\text{id}_B, n}(\eta_B, \tau) = \Sigma_{\text{id}_B} \Delta_{\text{id}_B}(\tau) \circ \eta_B = \tau \circ \text{id}_{\iota_B} = \tau$$

and

$$\mu_{m, \text{id}_B}(\sigma, \eta_B) = \Sigma_a \Delta_b(\eta_B) \circ \sigma = \Sigma_a \Delta_b(\text{id}_{\iota_B}) \circ \sigma = \Sigma_a(\text{id}_{\iota_E}) \circ \sigma = \text{id}_{\Sigma_a \iota_E} \circ \sigma = \sigma$$

as required. $\square$

B Factor graphs

B.1 Constructing the horizontal decorations

The apex decoration has a peculiar form, with one category of decorating data (the interfaces) further decorated by a family of others (the factors). This is because the apex decorations are themselves obtained by a Grothendieck construction, defined functorially for each finite set. The first step in this construction is to define the interface categories. Since we have the cospans determining which parts of the interfaces are exposed for factor composition, there may now be some parts which remain ‘latent’, and transformations of these parts are not constrained to be comonoid homomorphisms. That is, we can allow the transformations of the latent parts of the interfaces to be arbitrary morphisms in $\mathcal{C}$, as long as the exposed parts remain constrained as before⁵.

This line of thought formalizes as follows. Given a cospan $m : (A \xrightarrow{a} X \xleftarrow{b} B)$, we first define an enlarged version of F_0X , not totally restricted to comonoid homomorphisms. We denote this enlargement by $\tilde{F}_0m$ and construct it as the following limit of categories, so that we do still have comonoid homomorphisms over the legs:

$$\begin{array}{ccccc}
 \mathbf{Cat}(\text{disc}A, \mathbf{Comon}(\mathcal{C})) & \xleftarrow{\quad \tilde{F}_0m \quad} & & \xrightarrow{\quad} & \mathbf{Cat}(\text{disc}B, \mathbf{Comon}(\mathcal{C})) \\
 \downarrow & & \downarrow & & \downarrow \\
 \mathbf{Cat}(\text{disc}A, \mathcal{C}) & \xleftarrow{\quad \mathbf{Cat}(\text{disc}a, \mathcal{C}) \quad} & \mathbf{Cat}(\text{disc}X, \mathcal{C}) & \xrightarrow{\quad \mathbf{Cat}(\text{disc}b, \mathcal{C}) \quad} & \mathbf{Cat}(\text{disc}B, \mathcal{C})
 \end{array}$$

Note that, because the left and right vertical functors in this diagram are embeddings, so is the induced functor $\tilde{F}_0m \rightarrow \mathbf{Cat}(\text{disc}X, \mathcal{C})$. We can therefore understand $\tilde{F}_0m$ as the wide subcategory of $\mathbf{Cat}(\text{disc}X, \mathcal{C})$ whose morphisms are those X -indexed families of morphisms of $\mathcal{C}$ whose A - and B -components are comonoid homomorphisms.

The next step is to decorate $\tilde{F}_0m$ with factors. Since $\mathcal{C}$ is a category, factors on an interface $\chi : \text{disc}X \rightarrow \mathcal{C}$ form a set $\mathcal{C}(\chi^\otimes, I)$. We thus define a presheaf $\underline{F}_1m : \tilde{F}_0m^{\text{op}} \rightarrow \mathbf{Set}$ as follows. On objects (interfaces) χ , we define $\underline{F}_1m(\chi)$ to be $\mathcal{C}(\chi^\otimes, I)$. On morphisms of interfaces $\varphi : \chi \rightarrow \chi'$, we set $\underline{F}_1m(\varphi) : \underline{F}_1m(\chi') \rightarrow \underline{F}_1m(\chi)$ to be the precomposition functor $\mathcal{C}(\varphi^\otimes, I) : \mathcal{C}(\chi'^\otimes, I) \rightarrow \mathcal{C}(\chi^\otimes, I)$ and denote it by φ^* . This yields a well-defined presheaf by the functoriality of $\otimes$ and of precomposition. The apex decoration returned by F_1 on m is then given by the 1-categorical Grothendieck construction of $\underline{F}_1m$ (its category of elements), denoted $\int \underline{F}_1m$, whose objects and morphisms are as described in the main text.

This tells us how to decorate the apices of cospans — but, given the cospan m , F_1 needs to return a whole span of categories $F_0A \xleftarrow{\pi_a} \int \underline{F}_1m \xrightarrow{\pi_b} F_0B$. We obtain the legs of this span through the universal properties of the construction of $\int \underline{F}_1m$. First, because $\int \underline{F}_1m$ is obtained by a Grothendieck construction, it is equipped with a projection functor $\pi_m : \int \underline{F}_1m \rightarrow \tilde{F}_0m$ which acts by ‘forgetting’ the factors; *i.e.* $\pi_m(\chi, f) = \chi$. Next, the construction of $\tilde{F}_0m$ as a limit means that it is equipped with canonical projection functors to F_0A and F_0B :

$$F_0A = \mathbf{Cat}(\text{disc}A, \mathbf{Comon}(\mathcal{C})) \xleftarrow{\quad} \tilde{F}_0m \xrightarrow{\quad} \mathbf{Cat}(\text{disc}B, \mathbf{Comon}(\mathcal{C})) = F_0B$$

⁵It is this part of the construction that means we cannot use Fong’s original decorated cospans: here, the decoration on the apex necessarily depends on the legs, in order that we have comonoid homomorphisms in the right places.

Therefore, we obtain the requisite span of categories $F_1 m := (F_0 A \xleftarrow{\pi_a} \int \underline{F_1} m \xrightarrow{\pi_b} F_0 B)$ as the composition of this span of limit projections with π_m :

$$\begin{array}{ccccc}
 & & \int \underline{F_1} m & & \\
 & \swarrow \pi_a & \downarrow \pi_m & \searrow \pi_b & \\
 F_0 A & \xleftarrow{\quad} & \tilde{F}_0 m & \xrightarrow{\quad} & F_0 B
 \end{array}$$

We now need to define the action of F_1 on morphisms of cospans. Suppose that $m' := (A' \xleftarrow{a'} X' \xrightarrow{b'} B')$ is another cospan. A morphism $\phi : m \rightarrow m'$ is a triple (ϕ_l, ϕ, ϕ_r) of maps of finite sets making the following diagram commute:

$$\begin{array}{ccccc}
 A & \xrightarrow{a} & X & \xleftarrow{b} & B \\
 \phi_l \downarrow & & \downarrow \phi & & \downarrow \phi_r \\
 A' & \xrightarrow{a'} & X' & \xleftarrow{b'} & B'
 \end{array}$$

We know from the treatment above that F_0 is functorial, and this yields the action of F_1 on the left and right components ϕ_l and ϕ_r . Moreover, the construction of $\int \underline{F_1} m$ is functorial: first, the construction of $\tilde{F}_0 m$ is functorial in m , by the functoriality of taking limits. Likewise, the Grothendieck construction is functorial, yielding a functor $\int \underline{F_1} : \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} \rightarrow \mathbf{Cat}$. The following diagram then commutes by construction:

$$\begin{array}{ccccc}
 F_0 A & \xleftarrow{\pi_a} & \int \underline{F_1} m & \xrightarrow{\pi_b} & F_0 B \\
 F_0 \phi_l \downarrow & & \downarrow \int \underline{F_1} \phi & & \downarrow F_0 \phi_r \\
 F_0 A' & \xleftarrow{\pi_{a'}} & \int \underline{F_1} m' & \xrightarrow{\pi_{b'}} & F_0 B'
 \end{array}$$

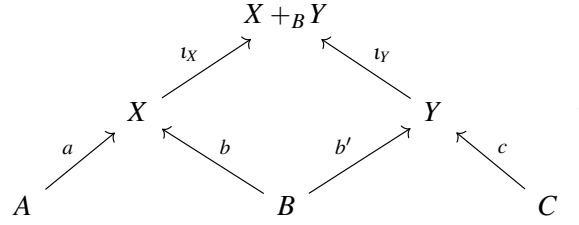
Consequently, we define the action of F_1 on the morphism of cospans ϕ to return the morphism of spans $F_1 \phi$ whose components are $(F_0 \phi_l, \int \underline{F_1} \phi, F_0 \phi_r)$, and we see that this action is functorial by construction.

B.2 The horizontal composition

μ must be a pseudo natural transformation filling the following diagram:

$$\begin{array}{ccc}
 \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} \times_{\mathbf{FinSet}} \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} & \xrightarrow{\diamond \mathbf{Cospans}(\mathbf{FinSet})} & \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} \\
 F_1 \times F_1 \downarrow & \nearrow \mu & \downarrow F_1 \\
 \mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}} \times_{\mathbf{Cat}} \mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}} & \xrightarrow{\diamond \mathbf{Span}(\mathbf{Cat})} & \mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}}
 \end{array}$$

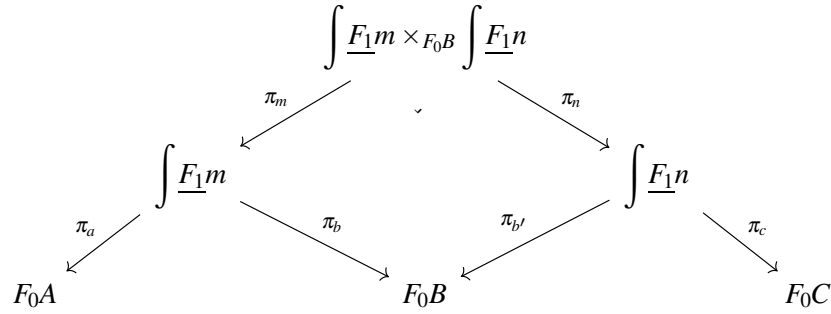
The external composition $\diamond_{\mathbf{Cosp}(\mathbf{FinSet})}$ acts on composable cospans by pushout, taking $m : (A \xrightarrow{a} X \xleftarrow{b} B)$ and $n : (B \xrightarrow{b'} Y \xleftarrow{c} C)$ to the outer cospan in the pushout diagram



(Pseudo functoriality of $\diamond_{\mathbf{Cosp}(\mathbf{FinSet})}$ follows from the universal properties of colimits.) The composite pseudo functor along the top and right of the above diagram therefore acts on m and n to yield the span of categories

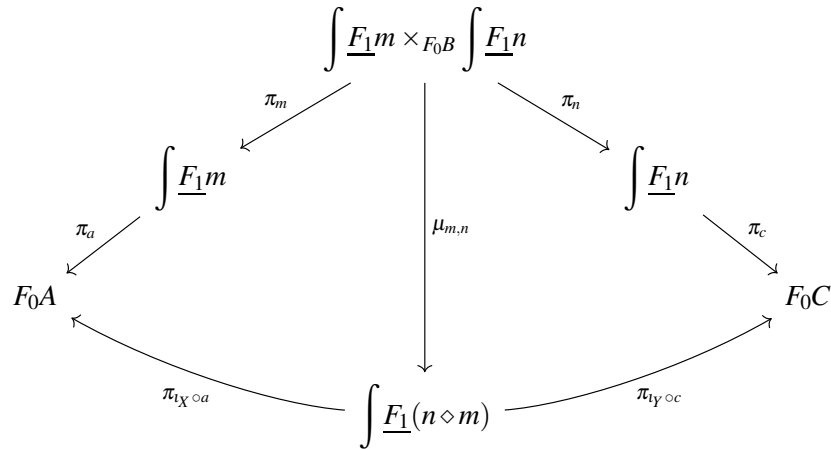
$$F_0 A \xleftarrow{\pi_{i_X \circ a}} \int \underline{F}_1(n \diamond m) \xrightarrow{\pi_{i_Y \circ c}} F_0 C .$$

Dually, $\diamond_{\mathbf{Span}(\mathbf{Cat})}$ acts on composable spans by pullback, so that the composite pseudo functor along the left and bottom of the diagram acts to yield the outer span in the following diagram:



Note that, here and throughout this section, π_m and π_n are the projections out of the pullback, rather than the Grothendieck fibrations of the previous section.

In both cases, the feet of the span are $F_0 A$ and $F_0 C$ on the left and right respectively; this must be the case for F to be a well-defined double functor. Consequently, the component of μ at (m, n) , denoted $\mu_{m,n}$, must be a functor $\int \underline{F}_1 m \times_{F_0 B} \int \underline{F}_1 n \rightarrow \int \underline{F}_1(n \diamond m)$ making this diagram commute:



This means that $\mu_{m,n}$ takes a pair of factors over X and Y , that are composable by agreeing over B , and returns a composite factor over $X +_B Y$, all while preserving the exposed interfaces over A and C . We define $\mu_{m,n}$ to act by copy-composition, as follows.

First, notice that both $\int \underline{F_1}(n \diamond m)$ and $\int \underline{F_1}m \times_{F_0B} \int \underline{F_1}n$ are (discrete) fibrations; the former over $\tilde{F}_0(n \diamond m)$ by definition and the latter over $\tilde{F}_0m \times_{F_0B} \tilde{F}_0n$, by the universal property of the pullback:

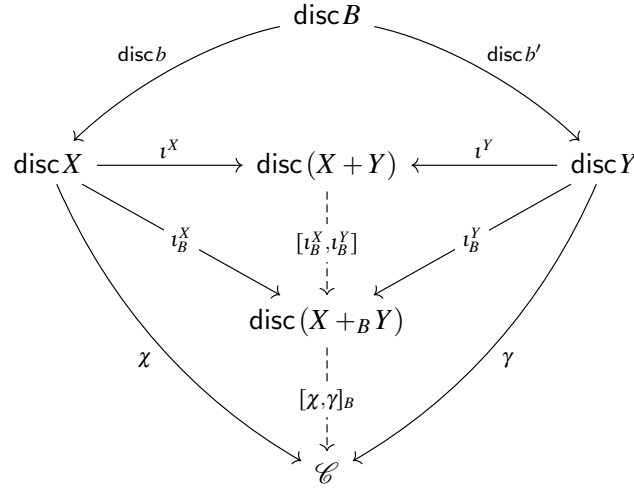
$$\begin{array}{ccccc}
 & \int \underline{F_1}m \times_{F_0B} \int \underline{F_1}n & & & \\
 & \swarrow & \downarrow \text{dashed} & \searrow & \\
 \int \underline{F_1}m & & \tilde{F}_0m \times_{F_0B} \tilde{F}_0n & & \int \underline{F_1}n \\
 \downarrow \pi_m & & \downarrow & & \downarrow \pi_n \\
 \tilde{F}_0m & & \checkmark & & \tilde{F}_0n \\
 & \swarrow & & \searrow & \\
 & F_0B & & &
 \end{array}$$

$\mu_{m,n}$ should therefore correspond to a morphism of fibrations. By the Grothendieck construction [16, p.7], such a morphism is equivalent to a pair $(\mu_{m,n}^0, \mu_{m,n}^1)$ of a functor and a natural transformation as in the following diagram:

$$\begin{array}{ccc}
 (\tilde{F}_0m \times_{F_0B} \tilde{F}_0n)^{\text{op}} & \xrightarrow{\mu_{m,n}^0} & \tilde{F}_0(n \diamond m)^{\text{op}} \\
 \searrow & \xRightarrow{\mu_{m,n}^1} & \swarrow \\
 (\underline{F_1}m, \underline{F_1}n) & & \underline{F_1}(n \diamond m) \\
 & \text{Set} &
 \end{array}$$

To define the functor $\mu_{m,n}^0 : \tilde{F}_0m \times_{F_0B} \tilde{F}_0n \rightarrow \tilde{F}_0(n \diamond m)$, note that the objects in its domain are pairs (χ, γ) of functors $\chi : \text{disc}X \rightarrow \mathcal{C}$ and $\gamma : \text{disc}Y \rightarrow \mathcal{C}$ that agree on B . By the universal properties of the coproduct $X + Y$ and the pushout $X +_B Y$, there must be copairings $[\chi, \gamma] : \text{disc}(X + Y) \rightarrow \mathcal{C}$ and $[\chi, \gamma]_B : \text{disc}(X +_B Y) \rightarrow \mathcal{C}$ making the following diagram commute; in particular, $[\chi, \gamma]$ must factor

through $[\chi, \gamma]_B$:



Since the image of (χ, γ) under $\mu_{m,n}^0$ must be a functor $\text{disc}(X +_B Y) \rightarrow \mathcal{C}$, we make the universal definition $\mu_{m,n}^0(\chi, \gamma) := [\chi, \gamma]_B$. Functoriality follows from the functoriality of colimits.

The corresponding component of $\mu_{m,n}^1$ must then be a function

$$\mu_{m,n}^1(\chi, \gamma) : \mathcal{C}(\chi^\otimes, I) \times \mathcal{C}(\gamma^\otimes, I) \rightarrow \mathcal{C}([\chi, \gamma]_B^\otimes, I).$$

Note that $[\chi, \gamma]^\otimes \cong \chi^\otimes \otimes \gamma^\otimes$. We can thus factor $\mu_{m,n}^1$ as

$$\mathcal{C}(\chi^\otimes, I) \times \mathcal{C}(\gamma^\otimes, I) \xrightarrow{\otimes} \mathcal{C}(\chi^\otimes \otimes \gamma^\otimes, I) \xrightarrow{\sim} \mathcal{C}([\chi, \gamma]^\otimes, I) \rightarrow \mathcal{C}([\chi, \gamma]_B^\otimes, I)$$

so that, by Yoneda, we seek a morphism $\delta_{\chi, \gamma} : [\chi, \gamma]_B^\otimes \rightarrow [\chi, \gamma]^\otimes$ in $\mathcal{C}$.

It is here that copying finally enters. For each $j : X +_B Y$, copying induces a morphism

$$\delta_{\chi, \gamma}^j : [\chi, \gamma]_B(j) \rightarrow \bigotimes_{i : [\iota_B^X, \iota_B^Y]^{-1}(j)} [\chi, \gamma](i)$$

in $\mathcal{C}$ which is unique up to coassociativity. Note that the codomain of this morphism is, by the pullback constraint, the monoidal product of $[\iota_B^X, \iota_B^Y]^{-1}(j)$ -many copies of the same object $[\chi, \gamma]_B(j)$ so that this copying is well-defined; and when j is not in the image of B , then this product is only unary, so that in this case, $\delta_{\chi, \gamma}^j$ is the corresponding identity morphism. Then

$$\bigotimes_{j : X +_B Y} \delta_{\chi, \gamma}^j : \bigotimes_{j : X +_B Y} [\chi, \gamma]_B(j) \rightarrow \bigotimes_{j : X +_B Y} \bigotimes_{i : [\iota_B^X, \iota_B^Y]^{-1}(j)} [\chi, \gamma](i)$$

has the requisite type, since $\bigotimes_{j : X +_B Y} [\chi, \gamma]_B(j) = [\chi, \gamma]_B^\otimes$ and

$$\bigotimes_{j : X +_B Y} \bigotimes_{i : [\iota_B^X, \iota_B^Y]^{-1}(j)} [\chi, \gamma](i) = \bigotimes_{i : X + Y} [\chi, \gamma](i) = [\chi, \gamma]^\otimes$$

both by definition. Hence we define $\delta_{\chi, \gamma} := \bigotimes_{j : X +_B Y} \delta_{\chi, \gamma}^j$.

We need to verify that $\delta_{\chi, \gamma}$ is natural in χ and γ . Since identity morphisms are always natural, we only need to check the case that $[\iota_B^X, \iota_B^Y]^{-1}(j)$ has more than one element — and this in turn only obtains

for j in the image of B . By construction, morphisms of interfaces on those components are necessarily comonoid homomorphisms, and these are the morphisms for which $\delta_{\chi,\gamma}^j$ is natural. Therefore, $\delta_{\chi,\gamma}$ is always natural in χ and γ , and since every factor of $\mu_{m,n}^1$ is accordingly natural, so is $\mu_{m,n}^1$ itself, as required.

This means that $(\mu_{m,n}^0, \mu_{m,n}^1)$ constitutes a well-defined morphism of indexed sets, thereby inducing a well-defined morphism $\mu_{m,n}$ of discrete fibrations. Thus, given factors (χ, f) and (γ, g) that agree over B , the external composition $\mu_{m,n}$ acts to return $([\chi, \gamma]_B, (f \otimes g) \circ \delta_{\chi,\gamma})$.

We also need to define the unit η , a pseudo natural transformation as in the following diagram:

$$\begin{array}{ccc}
 \mathbf{FinSet} & \xrightarrow{\text{id}_{\text{Cospans}(\mathbf{FinSet})}} & \mathbf{FinSet}^{\{\bullet \rightarrow \bullet \leftarrow \bullet\}} \\
 F_0 \downarrow & \nearrow \eta & \downarrow F_1 \\
 \mathbf{Cat} & \xrightarrow{\text{id}_{\text{Span}(\mathbf{Cat})}} & \mathbf{Cat}^{\{\bullet \leftarrow \bullet \rightarrow \bullet\}}
 \end{array}$$

The component η_X at each finite set X is determined by a functor $\eta_X : F_0X \rightarrow \int \underline{F_1}(\text{id}_X)$ as in the diagram

$$\begin{array}{ccccc}
 F_0X & \xlongequal{\quad} & F_0X & \xlongequal{\quad} & F_0X \\
 \parallel & & \downarrow \eta_X & & \parallel \\
 F_0X & \xleftarrow{\pi_X} & \int \underline{F_1}(\text{id}_X) & \xrightarrow{\pi_X} & F_0X
 \end{array}$$

where id_X denotes the identity cospan $X = X = X$ on X . Note that because $\int \underline{F_1}(\text{id}_X)$ is defined over the identity span, $\tilde{F}_0(\text{id}_X) \cong F_0X$, and so π_X simply forgets the factor decorations. Moreover, this means that all components of morphisms of interfaces in $\tilde{F}_0(\text{id}_X)$ are comonoid homomorphisms in $\mathcal{C}$.

We define η_X using the counits of the comonoid structures on the interfaces on X . That is to say, η_X maps an interface $\chi : \text{disc}X \rightarrow \mathbf{Comon}(\mathcal{C})$ to the pair (χ, ε_χ) , where here ε_χ denotes the canonical discarding map $\chi^\otimes \rightarrow I$. The functoriality of η_X follows from the fact that all morphisms in F_0X are comonoid homomorphisms, so if φ is a morphism $\chi \rightarrow \chi'$ then $\varepsilon_\chi = \varphi^* \varepsilon_{\chi'}$. Pseudo naturality of η obtains likewise, and its unitality with respect to μ follows from the counitality of the comonoid counits.

At this final point, we should also validate the remaining laws that a lax double functor should satisfy: in particular, the associativity of μ . However, we shall at this point simply assert that associativity follows from the associativity of $\otimes$, the coassociativity of the relevant comonoid structures, and the universal properties of the (co)limits involved in the constructions, leaving a detailed examination of this claim to future exposition.

B.3 Graphical calculus

The relationship between decorated cospans and undirected wiring diagram algebras indicates that factor graphs should be understood as composing ‘operadically’ (multicategorically): that is to say, by nesting; thus, within each factor of a factor graph may be hiding a whole other factor graph. This formalizes the situation sometimes encountered in the literature — *e.g.* Koudahl, van de Laar, and de Vries [15, Fig.20] — in which a collection of factors is grouped into a composite factor, sometimes depicted by drawing an extra box around the collection.

To hew more closely to existing undirected wiring diagrams, as well as the standard string diagrams for monoidal categories, we will adopt a syntax in which factors are depicted with circular boxes,

exposed variables by emanating wires, and transformation morphisms by rectangular boxes. We will depict comonoid homomorphisms as ‘diamond’ shaped boxes, both to indicate that they commute with horizontal composition $\diamond$ and also to suggest that they can pierce through the bubble enclosing a factor; this means we will attach wires to their corners. Conversely, we will depict arbitrary morphisms with standard rectangular boxes, attaching wires to their sides.

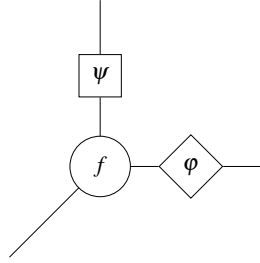
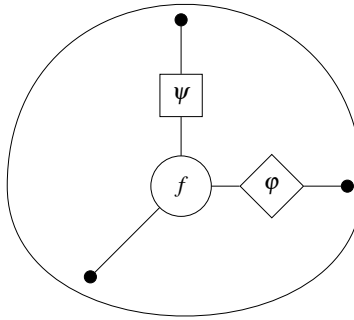


Figure 2: A composite factor f' .

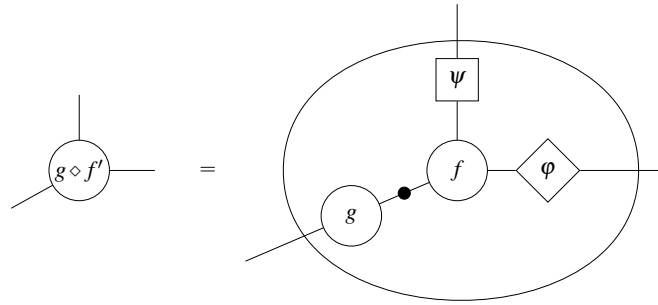
Figure 2 shows a simple example of a composite factor f' , constituted by another factor f with three exposed variables, two of which are composed with transformations, one deterministic (φ) and one not (ψ). Thinking statistically, we can interpret exposed variables (dangling wires) as representing *observed random variables*. We can draw a bubble around the whole of Figure 2 to obtain a new factor with three unobserved variables; nonetheless these variables are *observable*, which we depict by terminating them with a black dot. As a horizontal 1-cell in $\mathbb{FG}$, the resulting factor has the type $f_0 : 0 \rightarrow 0$, where 0 denotes the empty set.



The drawing of the bubble represents a 2-cell $f_0 \Rightarrow f'$. We can think of this 2-cell, being directed from f_0 to f' , as "reaching into f_0 ", pulling observable wires out to expose them as the wires of f' .

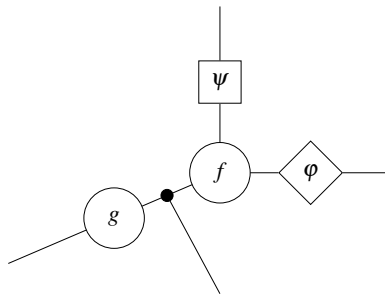
We can compose f with some factor g (with a compatible interface), repurposing the black dot — or

spider, in the terminology of Coecke and Kissinger [4] — to represent the locus of composition:

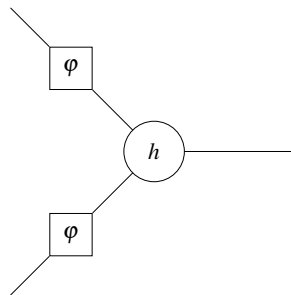


(In this way, we can think of a one-legged spider, representing an observable variable, as a locus of *potential* composition.)

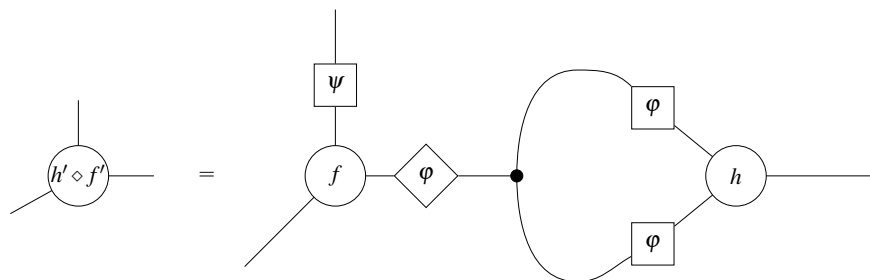
We can reach inside $g \diamond f'$ to expose the internal wire:



And this gives us another way to understand the deterministic morphisms: as those that can slide past spiders, multiplying on the way into factors. For example, suppose have another composite factor h' involving φ , such as this:

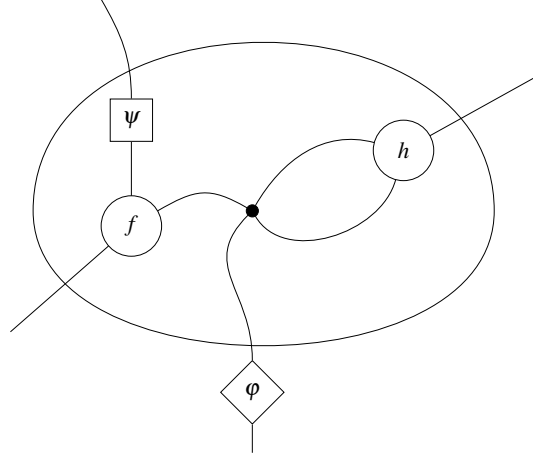


Then we could compose along φ with f' to obtain $h' \diamond f'$:



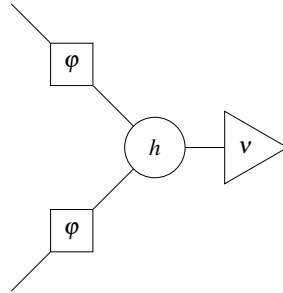
Notice that here we have made use of the ability (but not necessity!) of copy-composition to couple multiple variables together.

If again we expose the composed variable, we can pull φ outside, too:



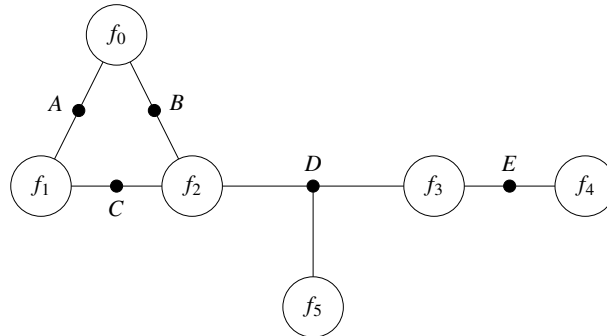
Notice that, by using spiders, there would be no ambiguity if we omitted the outer bubble: observed variables are those that are not terminated in a one-legged spider; and deterministic transformations “push forward” from exposed variables to multiply through spiders in the direction of their target factors.

We can use non-deterministic morphisms to render observable variables *unobservable*, by ‘marginalizing’ them out. For instance, here we have marginalized out one of the observed variables of h' by transforming it with a state v :



Here, we have used the standard string-diagrammatic representation of a state in $\mathcal{C}$.

Finally, let us depict the factor graph of Figure 1 in this language:



Note how similar this depiction is to the non-compositional form of Figure 1: the major change is the replacement of ‘variable’ nodes with (labelled) two-legged spiders, making the distinction between

observed and observable variables. Beyond this distinction, the syntax of $\mathbb{FG}$ tells us precisely what kinds of compositions are allowed — including supplying notions of *transformation* of factors.

Representing Knowledge and Querying Data using Double-Functorial Semantics

Michael Lambert

University of Massachusetts-Boston
michael.james.lambert@gmail.com

Evan Patterson

Topos Institute
evan@epatters.org

Category theory offers a mathematical foundation for knowledge representation and database systems. Popular existing approaches model a database instance as a functor into the category of sets and functions, or as a 2-functor into the 2-category of sets, relations, and implications. The functional and relational models are unified by double functors into the double category of sets, functions, relations, and implications. In an accessible, example-driven style, we show that the abstract structure of a ‘double category of relations’ is a flexible and expressive language in which to represent knowledge, and we show how queries on data in the spirit of Codd’s relational algebra are captured by double-functorial semantics.

1 Introduction

Knowledge representation and databases are among the most successful applications of category theory, supported not only by several decades of theoretical development [6, 11, 14, 16] but also by substantial practical and industrial deployments [10, 17]. The general scheme is that the *ontology* or *database schema* is a small category, possibly equipped with extra structure, and *instance data* is a structure-preserving functor out of the schema into a category such as **Set**. Under this scheme, knowledge representation and databases are two sides of the same coin: knowledge representation emphasizes rich schemas and expressive schema languages (categorical structure), whereas databases center the data instances and transformations between them. We will pass seamlessly between the terminologies of the two fields, as the distinction is artificial.

In Spivak’s category-theoretic model of a database [14], a *schema* is nothing more than a small category and an *instance* is a set-valued functor. With an eye toward knowledge representation, Spivak and Kent explored a richer language [16], defining an *ontology log*, or *olog*, to be a finite-limit, finite-colimit sketch and an *instance* to be a set-valued model of the sketch. In both cases the language is “functional,” as arrows in the schema are interpreted as functions between sets. Since much technology in knowledge representation, including the Web Ontology Language (OWL), favors relations over functions, the last author proposed an alternative notion of *relational olog* [8], taking Carboni and Walters’ ‘*bicategories of relations*’ as the basic structure [2]. An ontology now becomes a small ‘bicategory of relations’ and an instance is a structure-preserving 2-functor from it into **Rel**. In other words, morphisms in the schema are now interpreted as relations; also, there are 2-cells interpreted as implications.

We introduce double-categorical ologs, taking the first author’s ‘*double categories of relations*’ as the foundational structure [7]. In outline, a *double olog* will be a small double category, assumed locally posetal for simplicity and equipped with certain extra structure, and an *instance* will be a structure-preserving double functor into **Rel**, the double category of sets, functions, relations, and implications. As an obvious first benefit, this approach recognizes that functions and relations are fundamental concepts and grants them both first-class status. From the perspective of categorical logic, a double olog possesses

all four elements of a logic fibered over a type theory [5]: its objects are types and its arrows are terms; fibered over those, its proarrows are predicates and its cells are judgments or implications. By contrast, a functional olog is missing predicates and judgments, whereas a relational olog is missing terms.

Using the rich structure of a ‘double category of relations’, queries on the instance data of a double olog can be formulated internally to the schema, rather than through external mappings. In Spivak’s minimal model of databases [14, §5.3], queries are regarded as particular cases of the adjoint triple of data migration functors induced by a schema mapping. This approach, while mathematically economical, is indirect and requires delicate analysis to reduce to SQL primitives [17]. By contrast, the basic operations of Codd’s relational algebra [3, §2.1]—permutations, projections, joins—appear directly as operations in a ‘double category of relations’, namely as restrictions or extensions along symmetries, projections, or diagonals. Thus, as we show in Section 3, queries can be formulated as abstract relations in the schema and then evaluated by the double functor defining the instance. As a slogan, we say that *querying is double-functorial semantics*.

In this paper, we will explain the features of double ologs in an accessible, example-driven style, but for the benefit of readers familiar with double category theory, we briefly state the main technical definitions. Further background can be found in Appendix A. A ‘**double category of relations**’ is a locally posetal cartesian equipment satisfying a discreteness or Frobenius axiom [7, Definition 3.2]. We define a **double olog**, or just an **olog** for short, to be a small ‘double category of relations’, which we sometimes also assume to have tabulators. An **instance** of a double olog $\mathbb{D}$ is a cartesian, strict double functor $\mathbb{D} \rightarrow \mathbb{R}el$, which must also preserve tabulators whenever $\mathbb{D}$ is assumed to have them.

Acknowledgments Patterson was supported by the Air Force Office of Scientific Research (AFOSR) Young Investigator Program (YIP) through Award FA9550-23-1-0133.

2 Representing Knowledge

The two fundamental ingredients of a functional olog [16] are concepts (aka, types or objects) and attributes (aka, arrows or morphisms), schematized for example as

$$\boxed{\text{Sam Carter}} \xrightarrow{\text{is a}} \boxed{\text{person}},$$

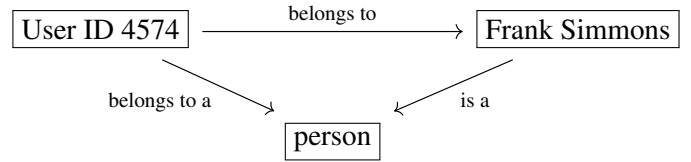
where the domain stands for a concept of *being Sam Carter*, the codomain represents the concept of *personhood*, and the arrow expresses that personhood is an attribute of Sam Carter.¹ This section will show how double categories—and in particular cartesian equipments and ‘double categories of relations’—can enhance the expressive power of ologs.

2.1 Expressing Facts

A **fact** in an olog is defined by Spivak and Kent [16] to be a commutative diagram. In other words, a fact is a statement that a pair of sequences of attributes have the same composite. Suppose we wish to assert

¹All of our examples are inspired by the television show *Stargate SG-1*, knowledge of which should enhance the reader’s enjoyment, if not their understanding, of the paper.

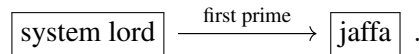
the fact that Frank Simmons is User 4574. In a functional olog, we could write



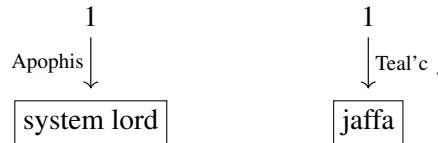
which says that Frank Simmons is the person to whom the ID ‘User 4574’ belongs.

However, something seems wrong about encoding individual people as types or objects in a categorical structure. Ordinarily, in first-order logic and type theory, individual entities are specified in a signature as constants of a certain type. In category theory, constants are interpreted as *global elements*, that is, as arrows $c: 1 \rightarrow A$ from a chosen terminal object to the type A . Following this approach, we would rather fix a terminal object and posit two individual constants.

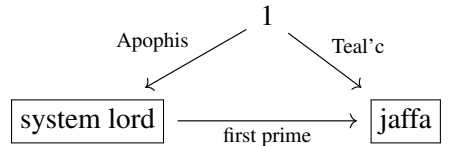
We start with an olog asserting that every System Lord has a First Prime, foremost among Jaffa warriors belonging to that System Lord. This could be expressed by a function



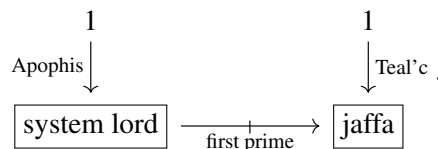
Let’s posit two individuals, Teal’c and Apophis, whom we include in the olog as constants



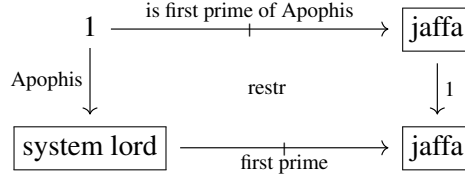
The fact that Teal’c is (or rather was) the First Prime of Apophis can be expressed as above by a commutative diagram



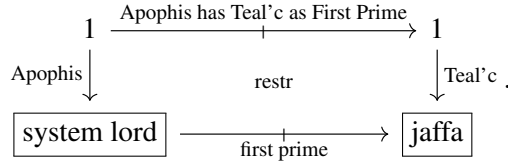
Now suppose we’d like to take a more nuanced view of the relationship “first prime.” The phrasing above carries the assumption that to each System Lord we can assign a unique First Prime, which is arguably true at a fixed moment in time. But what if at that moment in time a System Lord is between First Primes, so that in fact we have a partial function? Or perhaps we would like our data to include all of the First Primes a System Lord has ever had (a one-to-many relationship), or similarly we’d like to consider all of the System Lords a given Jaffa has served as First Prime. To accommodate these possibilities, we will alter the olog to specify a *relation* of being First Prime, along with two constants:



Restricting along the Apophis constant creates the cell



whose image under instance data valued in $\mathbb{R}el$ would amount to a list of all of Jaffa having served Apophis as first prime. How can we express the fact that Teal'c is (or was at some point) a first prime of Apophis? This is a matter of asking that another restriction factor through (or rather be equal to) the truth value $\top$, which is the terminal object in the external hom-category on 1. That is, we first form the restriction along both constants



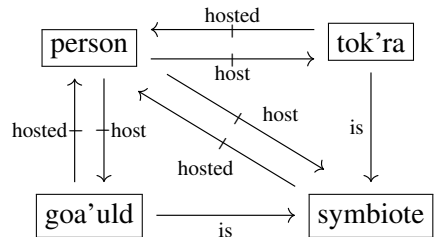
Denoting our double olog by $\mathbb{D}$, we have the truth value $\top : 1 \rightarrow 1$, the terminal object of $\mathbb{D}(1, 1)$. The fact that Apophis has (or had) Teal'c as a First Prime is just asking that the restricted proarrow above be equal to the local terminal:

$$1 \xrightarrow{\text{Apophis has Teal'c as First Prime}} 1 = \top .$$

This assertion cannot be so naturally expressed in either functional or relational ologs. In a functional olog, we must treat the relation of being First Prime as functional, which is too restrictive. On the other hand, in a relational olog, we can only introduce a constant through a relation $c : 1 \rightarrow A$ along with side equations making the relation into a *map* in the ‘bicategory of relations’ [2, Definition 1.5 and Lemma 2.5]. We then rely on the semantics of the olog in $\mathbf{Rel}$ to ensure that the relation c is interpreted as (the graph of) a function. In a double olog, we work natively with both functions and relations as is convenient.

2.2 Creating Types Using Tabulators

Here is another double olog, expressing a few relationships between persons, symbiotes, and types of symbiotes, namely, Goa'uld and Tok'ra:



Every person is a potential symbiote host. But notice that the relation $\text{host} : \boxed{\text{person}} \rightarrow \boxed{\text{symbiote}}$ and those valued in $\boxed{\text{tok'ra}}$ and $\boxed{\text{goa'uld}}$ are not functional since not every person is in fact a host. An

instance of “hosted” is generally many-to-one since many symbiotes have at least one host. It should be asked in equations or as a definition that $\text{host}^\dagger = \text{hosted}$, where the dagger $(-)^{\dagger}$ denotes the converse or opposite relation. All Tok’ra are symbiotes; so are Goa’uld. Thus, the two arrows labeled ‘is’ are genuinely functional, indicating something like subtypes.

Typically a Goa’uld leaves its host only upon the death of the host, and if the Goa’uld is forcibly removed, then the host will die due to release of toxins. So, simplifying slightly, we can make the assumption that each person is a host at most once. This is expressed by imposing the equation

$$\text{id} = \boxed{\text{goa'uld}} \xrightarrow{\text{hosted}} \boxed{\text{person}} \xrightarrow{\text{host}} \boxed{\text{goa'uld}}$$

asserting that the host and hosted relation compose to the identity on $\boxed{\text{goa'uld}}$. In other words, the relation host is a *partial map*. Theorem A.1 shows that if the double olog $\mathbb{D}$ were taken to be a unit-pure ‘double category of relations’ with tabulators, then this assumption would imply that the tabulator of the host-relation, namely $\boxed{\text{host-go'a'uld pairs}}$, is the apex of a span

$$\boxed{\text{person}} \xleftarrow{d} \boxed{\text{host-go'a'uld pairs}} \xrightarrow{c} \boxed{\text{goa'uld}}$$

where d is monic, that is, a partial map in the underlying category $\mathbb{D}_0$ of objects and arrows. If the tabulators in $\mathbb{D}$ are also assumed to be *strong*, then the process is reversible, in the sense that a partial map (a span with monic left leg) would induce a partial morphism in the relational sense. Thus, under suitable assumptions, one can pass back and forth between the two possible representations of a partial map in a double olog.

Even without these extra conditions, the mere presence of extensions and tabulators increases the expressive power of double ologs considerably. For example, we can create a type of persons who are or have been hosts. First, create a proposition for the notion *has hosted* as an extension cell

$$\begin{array}{ccc} \boxed{\text{person}} & \xrightarrow{\text{host}} & \boxed{\text{symbiote}} \\ \downarrow = & \text{ext} & \downarrow ! \\ \boxed{\text{person}} & \xrightarrow{\text{has hosted}} & 1 \end{array} .$$

This is interpreted as the image of the host relation in the type of persons. Then extract the comprehension or subobject classified by this proposition by taking its tabulator. This gives us a type of hosts:

$$\begin{array}{ccc} & \boxed{\text{host}} & \\ \swarrow \text{is} & & \searrow ! \\ \boxed{\text{person}} & \xrightarrow{\text{has hosted}} & 1 \end{array} \quad \text{tab}$$

Likewise, there is a type of hosted symbiotes constructed by the extension on the left followed by the tabulator on the right:

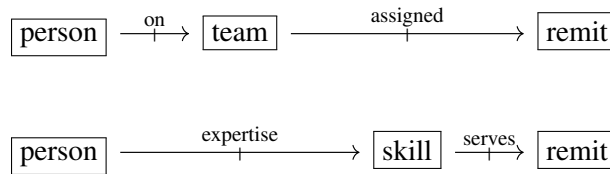
$$\begin{array}{ccc} \boxed{\text{person}} & \xrightarrow{\text{host}} & \boxed{\text{symbiote}} \\ \downarrow ! & \text{ext} & \downarrow = \\ 1 & \xrightarrow{\text{has been hosted}} & \boxed{\text{symbiote}} \end{array} \rightsquigarrow \begin{array}{ccc} & \boxed{\text{hosted symbiote}} & \\ \swarrow ! & & \searrow \text{is} \\ 1 & \xrightarrow{\text{has been hosted}} & \boxed{\text{symbiote}} \end{array} \quad \text{tab}$$

The arrow on the right again represents a kind of subtype since there are symbiotes such as queens who are not always hosted and symbiotes such as prim'ta who are never hosted for whatever reason. In short, tabulators enable propositions, and relations generally, to be reified as subtypes of the original types.

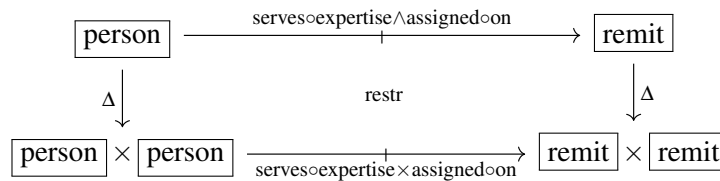
2.3 Forming Conjunctions Using Local Products

What if we want to assert as a fact that two propositions both hold or that they are equal? That can be done using *local products* (Appendix A.2).

Let's look again at an example. Consider the olog with four relations:



A fairly complex hypothetical underlies this olog, captured by the question of whether the expertise of given persons serves the remit of the teams of which they are members. We could require by fiat an equality between these objects in the appropriate hom-category. In some sense this *should be the case* if the teams have been formed correctly. But perhaps that's not necessarily the case, and we'd rather ask about *the extent to which* our question is answered in the affirmative. We can do this using a local product, which, in the example at hand, looks like



Let's postulate some concrete data to see what this local product looks like. Define a data instance $\mathbb{D} \rightarrow \mathbb{R}el$ for our double olog $\mathbb{D}$ by assigning the types to the data columns

person
Carter
Connor
Jackson
Kovacek
Teal'c

team
SG1
SG3
SG9
SG11

skill
archaeology
anthropology
astrophysics
engineering
law
philology
tracking
weapons

remit
combat
diplomacy
exploration
science & engineering
search & rescue

The first two relations are then the following:

on	
person	team
Carter	SG1
Connor	SG9
Jackson	SG1
Kovacek	SG9
Teal'c	SG1
Teal'c	SG3

assigned	
team	remit
SG1	exploration
SG3	combat
SG3	search & rescue
SG9	diplomacy
SG11	science & engineering

and the remaining two are the big tables filled with all of Daniel Jackson's PhDs:

expertise	
person	skill
Carter	astrophysics
Carter	weapons
Connor	engineering
Jackson	anthropology
Jackson	archaeology
Jackson	philology
Kovacek	law
Teal'c	weapons
Teal'c	tracking

serves	
skill	remit
astrophysics	exploration
anthropology	exploration
archaeology	exploration
engineering	science & engineering
law	diplomacy
philology	exploration
weapons	combat
weapons	exploration
tracking	search & rescue

None of these relations are functional. Perhaps some of the associations in the **serves** relation are debatable and certainly we haven't included all skills possessed by all the individuals in the **person** table, but the point is just to show how local products work. One can trace through the construction as a restriction first in $\mathbb{D}$ and then in $\mathbb{R}el$. But the local products formula in Equation (A.1) above makes the computation easy. The local product is just the set of all $\boxed{\text{person}} - \boxed{\text{remit}}$ pairs related under *both* composites $\text{serves} \circ \text{expertise}$ and $\text{assigned} \circ \text{on}$. To be explicit, we can compute the two composites as

serves $\circ$ expertise	
person	remit
Carter	exploration
Carter	combat
Connor	science & engineering
Jackson	exploration
Kovacek	diplomacy
Teal'c	combat
Teal'c	search & rescue

assigned $\circ$ on	
person	remit
Carter	exploration
Connor	diplomacy
Jackson	exploration
Kovacek	diplomacy
Teal'c	exploration
Teal'c	combat
Teal'c	search & rescue

In each case we've used the formula for composition of relations using existential quantification. Notice that despite Laurence Connor's training in aerospace engineering, he serves on the diplomacy team SG9, while Sam Carter's weapons training would qualify her for combat, but she is instead a member of

exploration team SG1. Thus, the local product is the table excluding these two entries, namely

$\text{serves} \circ \text{expertise} \wedge \text{assigned} \circ \text{on}$	
person	remit
Carter	exploration
Jackson	exploration
Kovacek	diplomacy
Teal'c	combat
Teal'c	search & rescue

The table thus gives precisely those people whose expertise serves the purpose of the teams to which they are assigned as well as the particular remit.

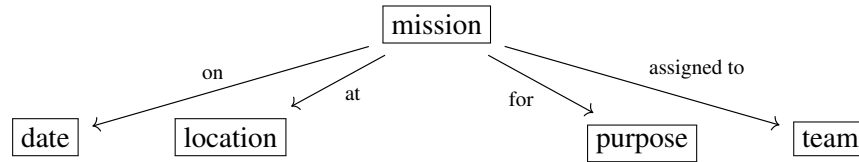
3 Querying Data

The conceit of this section is that ‘double categories of relations’ provide a simple and satisfying framework for querying data. In the approach based on functorial data migration [14], queries are handled externally through the device of adjoint functors between copresheaf categories and regarded as a special case of data migration. We will see here that queries can be expressed internally to the schema using the operations available in a ‘double category of relations’, or more generally in a cartesian equipment. As a slogan: *querying is double-functorial semantics*.

3.1 Select

The select operation is perhaps the easiest to describe in our double-categorical formalism. Selection is performed by taking an extension along projection morphisms.

Suppose we are describing the attributes belonging to a concept of a mission. In a functional olog [16], we would schematize the concept with functional relations indicated by the arrows identifying the leaf tables, such as:



However, this multi-span is just an encoding of a relation, which we choose to present double-categorically as a quaternary relation:

$$\boxed{\text{date}} \times \boxed{\text{location}} \xrightarrow{\text{mission}} \boxed{\text{purpose}} \times \boxed{\text{team}} .$$

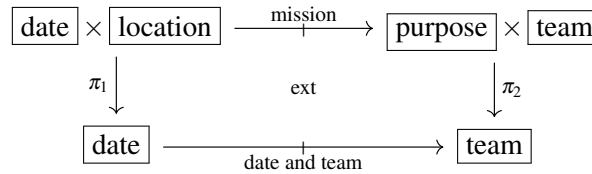
Of course, we could have chosen to partition the four types differently, but that would not meaningfully alter the result. We display the instance data for the relation in the conventional style as a multi-column

table:

mission			
date	location	purpose	team
02-06-1998	P41-771	search & rescue	SG3
07-31-1998	Cimmeria	assist Cimmerians	SG1
01-02-1999	P3R-272	investigate inscriptions	SG1
10-22-1999	Ne'tu	search & rescue	SG1
08-06-2004	Tegalus	negotiation	SG9

We'll assume in the background that the types have been instantiated with data, but it's not necessary to say at this point what the data actually is.

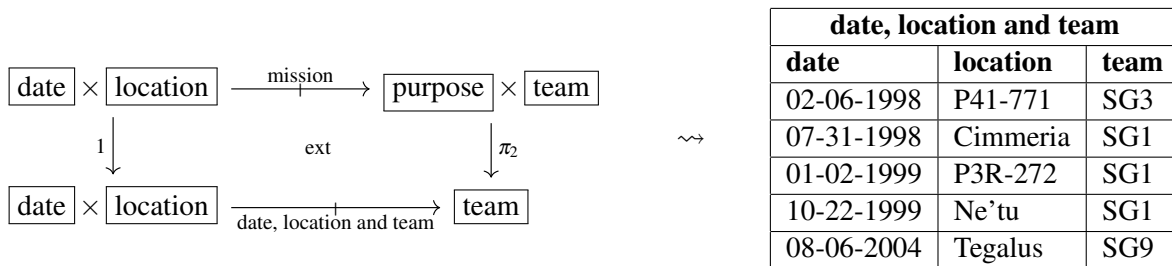
Selecting columns is now simply a matter of extending the relation along projections. For example, say we want only to remember the dates that teams went on missions but we care neither what the location nor purpose of the missions were. In this case, we take an extension along the projection morphisms as shown:



In the double category $\mathbb{R}el$, the extension is computed by taking an image. That is, a date and team are in the extension relation if, and only if, they are related (with two other elements of some 4-tuple) in the original relation. In this case, we end up with a binary relation instantiated by the table

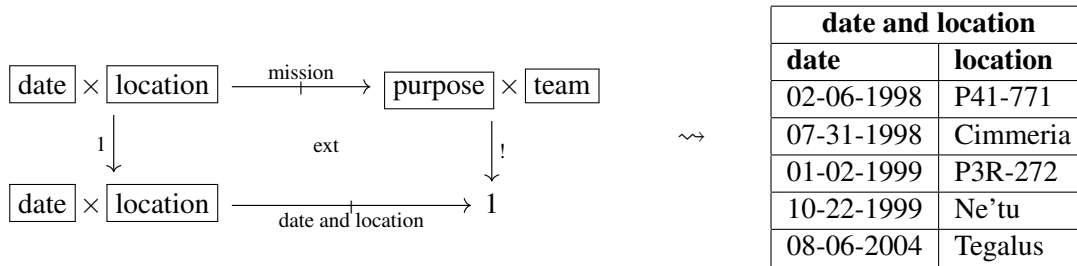
date and team	
date	team
02-06-1998	SG3
07-31-1998	SG1
01-02-1999	SG1
10-22-1999	SG1
08-06-2004	SG9

Alternatively, we might ask just for the occasions on which teams have visited locations disregarding the purpose of the mission. This would be accomplished by an extension yielding a ternary relation with accompanying table:



Finally, we might want to know only the date on which a location was visited. This would be an extension

using the terminal object on one side:



We take these examples to be sufficient evidence that the select operation can be performed by extending along projection morphisms. The point is simply to use whichever projections return the desired columns.

3.2 Filter

The operation of *filtering* data asks for rows that satisfy a certain property, in the simplest case that a particular attribute has a specific value. This is one of the last topics of [14], treated in §5.3 in an example using adjoint functors arising from slices of copresheaf toposes. In a double olog, filtering can be performed using restrictions provided we allow ourselves types in the olog isolating the attribute value that we're looking for.

Suppose we wish to filter the previous data for just the missions assigned to the flagship team SG1. As explained in Section 2.1, we can include the individual team SG1 in our olog in two different ways. First, we could add an arrow

$$\boxed{\text{SG1}} \xrightarrow{\text{is a}} \boxed{\text{team}} .$$

On the other hand, we could stipulate that SG1 is rather the name of a global element of the type, namely, an arrow from the terminal

$$1 \xrightarrow{\text{SG1}} \boxed{\text{team}} .$$

Both approaches suffice to filter the data but return slightly different arrangements. In the former case, we take the restriction

and the data returned is a subtable consisting only of certain rows of the original **mission** table, namely

SG1 missions			
date	location	purpose	team
07-31-1998	Cimmeria	assist Cimmerians	SG1
01-02-1999	P3R-272	investigate inscriptions	SG1
10-22-1999	Ne'tu	search & rescue	SG1

Of course, the last column on the right is superfluous since we are specifically filtering for that particular value. This redundancy can be avoided using the global element approach. In this case we have to be

especially careful to label the restricted relation in the olog. That is, we *define* the relation “SG1 missions” to be the restriction

$$\begin{array}{ccc}
 \boxed{\text{date}} \times \boxed{\text{location}} & \xrightarrow{\text{SG1 missions}} & \boxed{\text{purpose}} \times 1 \\
 \downarrow 1 & \text{restr} & \downarrow 1 \times \text{SG1} \\
 \boxed{\text{date}} \times \boxed{\text{location}} & \xrightarrow{\text{mission}} & \boxed{\text{purpose}} \times \boxed{\text{team}}
 \end{array}$$

This returns the simplified table omitting the redundant column:

SG1 missions		
date	location	purpose
07-31-1998	Cimmeria	assist Cimmerians
01-02-1999	P3R-272	investigate inscriptions
10-22-1999	Ne'tu	search & rescue

If in addition we wanted to select certain columns (say we were interested only in the dates on which SG1 visited certain places), we could use the projection-extension method above to produce a table with only those columns. Such a combination is effectively a *select-from-where* query in SQL, restricted to a single table. We now turn to how such queries can be extended to multiple tables.

3.3 Inner Joins

The inner join operation is an example of local products, already used in Section 2.3 and reviewed in Appendix A.2. Let's take a simple olog, namely, a fragment of the teams olog above with the two relations

$$\boxed{\text{person}} \xrightarrow{\text{expertise}} \boxed{\text{skill}} \qquad \boxed{\text{person}} \xrightarrow{\text{membership}} \boxed{\text{team}}$$

These relations are intended to express that a person may have expertise consisting of training or facility in a particular skill, and that each person may or may not belong to a particular SGC team. Denote this double olog by $\mathbb{D}$. As instance data $\mathbb{D} \rightarrow \mathbb{R}el$, let us assign the following data:

person
Hammond
Kovacek
Maybourne
Morrison
O'Neill
Rothman
Simmons
Warren

skill
archaeology
chicanery
combat
command
law
sociopathy

teams
SG1
SG3
SG9
SG11

expertise	
person	skill
Hammond	command
Kovacek	law
Maybourne	chicanery
Morrison	combat
O'Neill	command
Rothman	archaeology
Simmons	sociopathy
Warren	combat

membership	
person	team
Kovacek	SG9
Morrison	SG3
O'Neill	SG1
Rothman	SG11
Warren	SG3

As always, we assume that this data takes the form of a cartesian strict double functor $\mathbb{D} \rightarrow \mathbb{R}el$. In our example, the inner join along the shared column `person` is given by the restriction

$$\begin{array}{ccc}
 \boxed{\text{person}} & \xrightarrow{\text{expertise} \times \text{membership}} & \boxed{\text{skill}} \times \boxed{\text{team}} \\
 \Delta \downarrow & \text{restr} & \downarrow 1 \\
 \boxed{\text{person}} \times \boxed{\text{person}} & \xrightarrow{\text{expertise} \times \text{membership}} & \boxed{\text{skill}} \times \boxed{\text{team}}
 \end{array}$$

In relations, the restriction is computed as a pullback of the product along the arrow $\Delta \times 1$. That is, form the cartesian product of all pairs from the two relations and match the `person` argument. We obtain the following table

expertise $\bowtie$ membership		
person	skill	team
Kovacek	law	SG9
Morrison	combat	SG3
O'Neill	command	SG1
Rothman	archaeology	SG11
Warren	combat	SG3

recording both the skill and team of those persons who have both listing without any extra rows or null values. This is precisely the inner join of the two tables. Notice also that this approach avoids repeating the person column in the joined table.

4 Conclusion

We have shown by example how the structure of a ‘double category of relations’ can be used to represent knowledge in an ontology and to express queries against data instances of the ontology. Progressing from these illustrations to a working knowledge representation and database system will require further mathematical and engineering work.

Theoretical questions include what should be the basic double-categorical structure, including which notions are primitive and which are derived, and how the structure can be extended to meet practical needs. For instance, we have so far taken *local products* (conjunction) to be a concept derived from external products and restrictions, but in a different formulation of double-categorical products [9], local products become primitive operations, which is possibly more convenient. One unavoidable practical need is to support *data attributes* valued in fixed types such as strings or calendar dates. Several approaches to data attributes have been proposed in the literature on categorical databases [12, 14, 17]; at least one of them should be adapted to double-categorical databases. More ambitiously, it should be investigated whether our functorial approach to querying can be extended to support *aggregation* [15].

Design considerations for an engineered system have not been addressed. In an implementation, the mathematical expression of queries by double-functorial semantics must take the more concrete form of a query language, textual or graphical. Algorithms for evaluating queries in the language or reducing them to SQL queries must be devised. To do the latter, one would need to be more precise about the observation in Section 1 that the primitive notions of a ‘double category of relations’ closely resemble traditional relational algebra, in contrast to previous approaches to categorical databases. In this way, as always, engineering challenges may spur further theoretical development.

References

- [1] Evangelia Aleiferi (2018): *Cartesian double categories with an emphasis on characterizing spans*, PhD thesis. Dalhousie University, arXiv: 1809.06940.
- [2] Aurelio Carboni & Robert F.C. Walters (1987): *Cartesian bicategories I*. *Journal of Pure and Applied Algebra*. 49(1-2), pages 11–32, doi: 10.1016/0022-4049(87)90121-6.
- [3] Edgar F. Codd (1970): *A relational model of data for large shared data banks*. *Communications of the ACM*. 13(6), pages 377–387, doi: 10.1145/362384.362685.
- [4] Marco Grandis (2019): *Higher dimensional categories: From double to multiple categories*. World Scientific. doi: 10.1142/11406.
- [5] Bart Jacobs (1999): *Categorical logic and type theory*. Elsevier.
- [6] Michael Johnson, Robert Rosebrugh & R.J. Wood (2002): *Entity-relationship-attribute designs and sketches*. *Theory and Applications of Categories*. 10(3), pages 94–112. Available at <http://www.tac.mta.ca/tac/volumes/10/3/10-03abs.html>.
- [7] Michael Lambert (2022): *Double categories of relations*. *Theory and Applications of Categories*. 38(33), pages 1249–1283, arXiv: 2107.07621. Available at <http://www.tac.mta.ca/tac/volumes/38/33/38-33abs.html>.
- [8] Evan Patterson (2017): *Knowledge representation in bicategories of relations*, arXiv: 1706.00526.
- [9] Evan Patterson (2024): *Products in double categories, revisited*, arXiv: 2401.08990.
- [10] Evan Patterson, Owen Lynch & James Fairbanks (2022): *Categorical data structures for technical computing*. *Compositionality*. 4(5), doi: 10.32408/compositionality-4-5, arXiv: 2106.04703.
- [11] Frank Piessens & Eric Steegmans (1995): *Categorical data-specifications*. *Theory and Applications of Categories*. 1(8), pages 156–173. Available at <http://www.tac.mta.ca/tac/volumes/1995/n8/1-08abs.html>.
- [12] Patrick Schultz, David I. Spivak, Christina Vasilakopoulou & Ryan Wisnesky (2017): *Algebraic databases*. *Theory and Applications of Categories*. 32(16), pages 547–619, arXiv: 1602.03501.
- [13] Michael Shulman (2008): *Framed bicategories and monoidal fibrations*. *Theory and Applications of Categories*. 20(18), pages 650–738, arXiv: 0706.1286. Available at <http://www.tac.mta.ca/tac/volumes/20/18/20-18abs.html>.
- [14] David I. Spivak (2012): *Functorial data migration*. *Information and Computation*. 217, pages 31–51, doi: 10.1016/j.ic.2012.05.001, arXiv: 1009.1166.
- [15] David I. Spivak (2021): *Functorial aggregation*, arXiv: 2111.10968.
- [16] David I. Spivak & Robert E. Kent (2012): *Ologs: A categorical framework for knowledge representation*. *PloS One*. 7(1), e24274, doi: 10.1371/journal.pone.0024274, arXiv: 1102.1889.
- [17] David I. Spivak & Ryan Wisnesky (2015): *Relational foundations for functorial data migration*. In *Proceedings of the 15th Symposium on Database Programming Languages*, pages 21–28, doi: 10.1145/2815072.2815075, arXiv: 1212.5303.

A Background on Double Category Theory

In this appendix, we review background material on double category theory, as well as prove a relevant new characterization of partial maps in a ‘double category of relations.’ A general reference on double category theory is the textbook by Grandis [4].

A.1 Double Categories of Relations

Briefly, a ‘**double category of relations**’ [7] is a double category that is also cartesian and an equipment and that satisfies a discreteness condition distinguishing the ‘bicategories of relations’ of [2] from mere cartesian bicategories. For simplicity we also assume that any such ‘double category of relations’ is locally posetal. The usual double category of relations $\mathbb{R}el$ is the canonical example. That a ‘double category of relations’ $\mathbb{D}$ is cartesian means that it has finite products in an appropriately coherent way [1], namely, the canonical double functors $\mathbb{D} \rightarrow \mathbb{D} \times \mathbb{D}$ and $\mathbb{D} \rightarrow 1$ have right adjoints in the 2-category of double categories, (pseudo)-double functors and transformations. That a ‘double category of relations’ is an equipment [13] means that the canonical source-target projection functor $\mathbb{D}_1 \rightarrow \mathbb{D}_0 \times \mathbb{D}_0$ is a bifibration.

In more practical terms, the latter bifibration condition is equivalent to the fact that niches and coniches can be completed to proper cells

$$\begin{array}{ccc} A & \dashrightarrow & B \\ f \downarrow & \text{restr} & \downarrow g \\ C & \xrightarrow{S} & D \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{R} & B \\ h \downarrow & \text{ext} & \downarrow k \\ C & \dashrightarrow & D \end{array}$$

that are (respectively) cartesian and opcartesian with respect to the projection functor $\mathbb{D}_1 \rightarrow \mathbb{D}_0 \times \mathbb{D}_0$. The cartesian cell on the left is thought of as a *restriction* since in $\mathbb{R}el$ it is given by taking a pullback of the relation S along the pair $f \times g$. On the other hand, the opcartesian cell on the right is an *extension*, since in $\mathbb{R}el$ it would be given by taking an image.

Such restrictions and extensions, together with the product structure, do most of the querying work in Section 3. *Local products* make sense in this context and are discussed below in Appendix A.2. A ‘double category of relations’ has a further property, however, namely, that of satisfying a *discreteness axiom*, which ensures that its horizontal bicategory is in fact a compact closed monoidal category. This gives a good notion of partial map which is discussed below in Appendix A.3.

A double category $\mathbb{D}$ has **tabulators** if the external identity functor $\mathbb{D}_0 \rightarrow \mathbb{D}_1$ has a right adjoint $\top$. This associates to each proarrow an object and a universal cell that we think of as a kind of *comprehension scheme*. Tabulators provide type creation in double categorical ologs by associating a type to a proarrow of those entities which satisfy the proposition represented by the given proarrow.

Our data tables will be given by strict, cartesian double functors $M: \mathbb{D} \rightarrow \mathbb{R}el$ valued in the double category of relations. Strict means that the associators and unitors are not mere isomorphisms, but are in fact equalities. Pseudo, hence strict, double functors preserve restriction and extension cells [13, §6]. Strict cartesian double functors $M: \mathbb{D} \rightarrow \mathbb{R}el$ also preserve local products, as discussed immediately below. These preservation properties are used through Section 3 of the paper and those calculations support our conceit that querying is functorial semantics.

A.2 Local Products

Given two proarrows, say $R: X \rightarrowtail Y$ and $S: X \rightarrowtail Y$ in a cartesian equipment $\mathbb{D}$, the **local product** of R and S is defined to be the proarrow domain of the restriction along the diagonals

$$\begin{array}{ccc} X & \xrightarrow{R \wedge S} & Y \\ \Delta \downarrow & \text{restr} & \downarrow \Delta \\ X \times X & \xrightarrow{R \times S} & Y \times Y \end{array} .$$

It computes a binary product in the hom-category $\mathbb{D}(X, Y)$. It's pretty easy to see that in $\mathbb{R}el$, this local product is computed as a certain set of pairs. Namely, since it's a restriction, the resulting relation is the monic on the leftside of

$$\begin{array}{ccc} R \wedge S & \xrightarrow{\quad} & R \times S \\ \downarrow & & \downarrow \\ X \times Y & \xrightarrow{\Delta_X \times \Delta_Y} & X \times X \times Y \times Y \end{array}$$

where we've forgotten about one of the associativity isomorphisms in the product on the right. In any case, $R \wedge S$ in $\mathbb{R}el$ is the set of pairs

$$R \wedge S = \{(x, y) \mid xRy \text{ and } xSy\}. \quad (\text{A.1})$$

Thus, local products can be constructed by restricting products of relations along diagonal arrows. Alternatively, local products can viewed as a primitive notion with their own double-categorical universal property [9, Example 6.10], and any cartesian equipment possesses finite local products in this sense [9, Corollary 8.7].

Inner joins can be constructed by adapting local products, which involves matching along a diagonal. That is, given two proarrows $R: A \times B \rightarrow C$ and $S: B \rightarrow D$, we define the **inner join of R and S along B** to be the restriction cell

$$\begin{array}{ccc} A \times B & \xrightarrow{R \times S} & C \times D \\ 1 \times \Delta \downarrow & \text{restr} & \downarrow 1 \\ A \times B \times B & \xrightarrow{R \times S} & C \times D \end{array} \quad .$$

Think of this as giving two tables, namely, R and S with entries from sets A , B , C and D . The tables have entries in one column in common, namely, those from the set B . We take the cartesian product of the relations $R \times S$ and then pull back along the diagonal morphism Δ on B asking that those projection values are the same while leaving the others alone.

A.3 Partial Maps

Any ordinary relation $R: A \rightarrow B$ has a reverse relation $R^\dagger: B \rightarrow A$ which exchanges the domain and codomain. Suppose that $R: A \rightarrow B$ is a relation whose associated span

$$\begin{array}{ccc} R & \xrightarrow{c} & B \\ d \downarrow & & \\ A & & \end{array}$$

is a *partial map* in the sense that d is monic. Define a morphism $\varepsilon: R \times_A R \rightarrow B$ taking (r, s) in the pullback of d along itself to $cr = cs$, where equality is implied by the fact that d is monic. This map ε defines a cell

$$\begin{array}{ccc} R \times_A R & \xrightarrow{\langle c, c \rangle} & B \times B \\ \varepsilon \downarrow & (I) & \downarrow 1 \\ B & \xrightarrow{\Delta} & B \times B \end{array}$$

which amounts to the condition that $R^\dagger \otimes R \leq \text{id}_B$ holds in $\mathbb{R}el$.

The converse is true. Consider any fixed relation $R: A \rightarrowtail B$ and its reverse relation $R^\dagger: B \rightarrowtail A$. Suppose that $R^\dagger \otimes R \leq \text{id}_B$ holds. There is then a cell $\varepsilon: R^\dagger \otimes R \rightarrow B$ which is an arrow making a commutative square of the form of (I) above. The existence of ε forces d to be monic. For if $r = (x, y)$ and $s = (x, z)$ are given elements of $R \times_A R$ (which is the same as supposing that $dr = ds$), we have that

$$\Delta\varepsilon(r, s) = \langle c, c \rangle(r, s) = (y, z) \quad (\text{A.2})$$

is a diagonal entry of $B \times B$ by (I) above meaning that $y = z$ must hold. Thus, the span consisting of d and c is a genuine partial map $A \rightarrow B$.

A general version for the necessity of the condition is valid in any unit-pure (the external identity functor is fully faithful) ‘double category of relations’ with tabulators $\mathbb{D}$. Recall that in this case the horizontal bicategory $\mathbf{H}(\mathbb{D})$ is then a cartesian bicategory [7, Proposition 3.1] and indeed a $\mathbf{H}(\mathbb{D})$ is a ‘bicategory of relations’, hence a compact closed monoidal category with involution on proarrows denoted by $(-)^{\dagger}$ [2, Theorem 2.4]. Recall from [2, §2] that a proarrow $r: A \rightarrowtail B$ is a **partial map** if $r^\dagger \otimes r \leq \text{id}_B$ holds.

Theorem A.1. *If $\mathbb{D}$ is unit-pure and has tabulators, then $r: A \rightarrowtail B$ is a partial map only if $d: \top r \rightarrow A$ is a monic arrow, meaning, in other words, that the span formed by the legs of $\top r$ is a partial map in the sense that the domain arrow d is monic.*

Proof. Suppose that r is a partial map and suppose as given two morphisms $f, g: X \rightrightarrows \top r$ satisfying $df = dg$. Using the cell $r^\dagger \otimes r \leq \text{id}_B$ and “unit-pure” there is a unique morphism h that is equal to each of the legs in the outside of the figure on the left in

$$\begin{array}{ccc} \top r \times_A \top r & \longrightarrow & \top r \xrightarrow{c} B \\ \downarrow & \lrcorner & \downarrow d \\ \top r & \xrightarrow{d} & A \\ c \downarrow & & \\ B & & \end{array} \qquad \begin{array}{c} \top r \times_A \top r \\ \exists! h \downarrow \\ B \end{array}$$

in that $c\pi_1 = h = c\pi_2$ holds. Thus, f and g induce a pair morphism $\langle f, g \rangle: X \rightarrow \top r \times_A \top r$, and we can calculate that

$$cf = c\pi_1 \langle f, g \rangle = h \langle f, g \rangle = c\pi_2 \langle f, g \rangle = cg \quad (\text{A.3})$$

meaning that $f = g$ must hold as a result of the uniqueness clause of the universal property of the tabulator $\top r$. Therefore, d is monic. $\square$

A Category Theoretic Approach to Compositional Approximation in Game Theory

Neil Ghani

MSP Group, University of Strathclyde, Scotland
ng@cis.strath.ac.uk

This paper uses category theory to guide the development of an entirely new approach to approximate game theory. Game theory is the study of how different agents within a multi-agent system take decisions. At its core, game theory asks what an optimal decision is in a given scenario. Thus approximate game theory asks what is an approximately optimal decision in a given scenario. This is important in practice as — just like in much of computing — exact answers maybe too difficult (or even impossible) to compute given inherent uncertainty in input.

We consider first *Selection Functions* which are a simple model of compositional game theory. We develop i) a simple yet robust model of approximate equilibria; ii) the algebraic properties of approximation with respect to selection functions; and iii) relate approximation to the compositional structure of selection functions. We then repeat this process for *Open Games* — a more advanced model of game theory featuring the key operation of sequential composition of games. Finally, we use approximation to develop new metrics on game theory and show these yield core theorems in what one might term “Metric Game Theory”.

1 Introduction

This paper uses category theory to develop an entirely new approach to compositional approximate game theory. Approximation in computation is ubiquitous e.g. it arises in: i) stochastic systems, where one merely has probability distributions over values induced by inherent/simulated randomness; ii) resource limited environments, where exact computation is prohibitively expensive; iii) systems with imperfect/partial recall, where one only has limited information about what has happened or the intentions/trustworthiness of each agent; and iv) non-exact computation where primitive data (e.g. from sensors) is inexact and supplied with error bars. These scenarios arise in e.g. cyber-physical systems, machine learning, robotics, automotive engineering, aerospace, and energy systems.

On the other hand, game theory is the study of how different agents within a multi-agent system take decisions. The simplest model of game theory we consider are the *multi-valued selection functions* [5] which are functions $s : (X \rightarrow V) \rightarrow PX$. Henceforth we merely call them selection functions. Here, i) the set X is thought of as a set of actions (or moves, or choices) the agent can consider; ii) the set V is thought of as a type of utilities; and iii) a function $X \rightarrow V$ is called a *utility function* as it assigns, to each possible action, a utility measuring how good that action is deemed to be. A selection function then maps *every* utility function to those actions deemed optimal for that utility function as such is a model of game theory. The canonical selection function is argmax defined by

$$x \in \text{argmax } f \text{ iff } (\forall x' : X) f x \geq f x'$$

Another selection function is ε -argmax defined by

$$x \in \varepsilon\text{-argmax } f \text{ iff } (\forall x' : X) f x \geq f x' - \varepsilon$$

which means for an action x to be ε -optimal it need merely have an associated utility which is within ε of a true optimal action. This approach aligns with the general approach to approximation of uncertainty budgets as we track - and therefore can check - how much utility is lost by moving to an approximately optimal decision. Notice this is NOT saying x is close to an optimal choice of action, rather that the utility of x is close to being the utility of an optimal action. One might study the former and, indeed, this was our first intention. However, the mathematical behaviour of that concept was poorer so we switched to the approach taken in this paper. Nevertheless, this was the first of several moments where key decisions need to be taken to obtain a well behaved mathematical theory showing the combination of approximation and game theory to be a delicate and nuanced endeavour.

A second difficulty arises from the desire for *compositionality* to ensure scalability. Compositional systems are those where larger systems can be defined by assembling smaller subsystems together which means that proving properties of larger systems can be reduced to proving properties of smaller (and hence easier to reason about) subsystems. At its core, compositionality is a story about structure and so the natural mathematical medium in which to develop compositionality is category theory. Recently, Compositional Game Theory [9] was introduced to bring the advantages of compositionality to game theory. However, within compositional game theory there are a number of different notions of morphisms of open games and, as we shall see, certain notions behave better with regards to approximation than others. Finding the right notion of morphisms of open games to interact well with approximation was another issue we had to solve.

Concretely, this paper will

- introduce a notion of the ε -approximation of a game by defining for any game G , another game $T_\varepsilon G$ which we think of as containing not just the equilibria of G , but also the ε -equilibria of G — this is an entirely new approach to approximation of equilibria and shows novelty of methodology.
- show that T_ε has good algebraic structure in that it is i) functorial; and ii) possesses graded structure — this shows our methodology is canonical in being aligned with other work on quantitative computation.[2]
- show T_ε interacts well with the compositional structure of game theory, in particular with Nash equilibria and sequential composition. There is no particular reason that the approximation of a large game built from game-constructors should retain that structure — we show this holds for the key cases of parallel (or Nash) product and sequential composition. This is crucial evidence for the utility of our approach, namely that preserves the compositional structure of game theory.
- use our approximate game theory to create what one might call *Metric Game Theory* which i) not only builds new games from old; but ii) ensures if one substitutes a subgame with another which has similar behaviour, the overall system remains similar to the original. The crucial construction permitting *Metric Game Theory* is a metric on games which we build from our approximate game theory.

This third item is particularly surprising and thus our strongest contribution — as mentioned there is no inherent reason why, for example, the approximations of a Nash equilibria should form a Nash equilibria.

Finally, we argue in the conclusions that our methodology is also general in that it can apply to other forms of game theory, eg Markov Decision Processes, and beyond game theory into the huge area of machine learning. This latter application area is particularly exciting as formally quantifying distance from optimality is a way of expressing how much has been learned and thus contributing to the XAI agenda.

This paper is structured as follows. In Section 2, we cover preliminaries pertaining to metric spaces, lenses and selection functions. Section 3 develops approximation for selection functions, discusses the algebraic properties of approximation and shows approximation is compositional. Section 4 extends these results to open games [9], while Section 5 uses approximation to develop Metric Game Theory. Section 6 concludes the paper with conclusions and directions for future research.

2 Preliminaries: Metric Spaces and Lenses

This paper should be accessible to those with a basic understanding of game theory and category theory. Introductory texts are [18, 14, 16], for game theory and [15, 13] for category theory.

2.1 Metric Spaces

Let $\mathbb{R}$ be the positive reals extended with an infinity ∞

Definition 2.1. *Define:*

1. *An extended metric space is a set X with a function $d : X \times X \rightarrow \mathbb{R}$ satisfying symmetry, the triangle inequality and $d(x, y) = 0$ iff $x = y$.*
2. *short maps between metric spaces (X, d) and (Y, d') are functions $f : X \rightarrow Y$ such that $d(fx, fy) \leq d(x, y)$.*

We write $|X|$ for the underlying set of a metric space X .

Definition 2.2. *If (X, d) and (Y, d') are extended metric spaces, then $(X, d) \otimes (Y, d')$ is the extended metric space with underlying set $X \times Y$, where*

$$d_{X \otimes Y}((x, y), (x', y')) = \max\{d(x, x'), d'(y, y')\}$$

Other options are possible, eg sum, average etc. The following explains the use of ∞ as a possible distance — it is needed to ensure the metric on function spaces is well defined.

Definition 2.3. *Let X, Y be a metric spaces. The set of functions $[X, Y]$ is a metric space via the sup-metric:*

$$d(f, g) = \sup_{x \in X} d(fx, gx).$$

2.2 Lenses

Lenses underpin selection functions and open games and so we introduce them here. While lenses can be built over categories with appropriate structure, to keep our presentation simple, we work with lenses over sets.

Definition 2.4. *A lens is given by a pair of sets (X, V) . A map of lenses $f : (X, V) \rightarrow (Y, Z)$ is given by a pair (f_0, f_1) consisting of functions $f_0 : X \rightarrow Y$ and $f_1 : X \times Z \rightarrow V$.*

The following is well known [9, 11]:

Lemma 2.5. *Lenses from a symmetric monoidal category $\mathbf{Lens}$ with tensor given by $(X, V) \otimes (X', V') = (X \times X', V \times V')$ and with unit given by $I = (1, 1)$.*

Note i) an element of X is a lens-map $I \rightarrow (X, V)$; and ii) a function $X \rightarrow V$ is equivalently a lens map $(X, V) \rightarrow I$. These two observations are useful, eg we can often represent complex functions $X \rightarrow V$ more simply as composites of lens maps.

2.3 Selection Functions

A basic form of game theory is given by multi-valued selection functions [5] which, given sets X, Y , are defined by $\text{Sel}(X, Y) = (X \rightarrow Y) \rightarrow PX$ where PX is the set of powersets of X . We recall that for simplicity we just call them selection functions henceforth. Intuitively, a selection function maps a utility function (which assigns to each potential action $x \in X$ to a resulting utility $y \in Y$) to a set of actions deemed to be optimal for that specific utility function. Give two selection functions $s, t : (X \rightarrow V) \rightarrow PX$, we write $s \subseteq t$ iff $(\forall f : X \rightarrow V) sf \subseteq tf$. The canonical selection function is argmax defined as follows:

Example 2.6. *Define the argmax selection function $\text{argmax} : (X \rightarrow Y) \rightarrow PX$ by*

$$\text{argmax}f = \{ x \in X \mid (\forall x' \in X) fx \geq fx' \}$$

Note that the selection function model of game theory deals with not merely one utility function but, rather, all utility functions. This is also the case of open games and is a hallmark — indeed key distinguishing feature — of compositional models of game theory. When it comes to maps of selection functions, we choose the following for reasons that will become clearer later.

Definition 2.7. *Let $s : \text{Sel}(X, V)$ and $t : \text{Sel}(X', V')$. A map from s to t consists of a map of lenses $\alpha : (X, V) \rightarrow (X', V')$ such that*

$$(\forall k : X' \rightarrow V')(\forall x : X) \alpha x \in tk \Rightarrow x \in s(k\alpha)$$

Note in the above we write the map $X \rightarrow V$ as the lens-composite of lens maps $k\alpha$ as this is much cleaner than a direct definition. This is an one example of the power of lens composition to simplify definitions as we remarked earlier. We continue to use lens composition via juxtaposition through the rest of the paper without remarking on it. The above definition also possesses a contravariance akin to how maps of containers (aka dependent lenses) are contravariant in their position maps [1]. An alternative covariant definition for a map of selection functions would have been to require

$$(\forall k : X' \rightarrow V') x \in s(k\alpha) \Rightarrow \alpha x \in tk$$

but, as we shall see later, this does not give a functorial theory of approximation. Henceforth we write Sel for the category of all selection functions and the maps between them. Selection functions are a compositional model of game theory as they support a Nash product. This arises by extending the monoidal product on lenses to one on selection functions:

Definition 2.8. *The category Sel of selection functions and maps of selection functions is monoidal with monoidal product, also called the Nash product,*

$$\otimes : \text{Sel}(X, Y) \times \text{Sel}(X', Y') \rightarrow \text{Sel}(X \times X', Y \times Y')$$

defined by

$$(x, x') \in (s \otimes t)f \quad \text{iff} \quad \begin{aligned} & x \in s(\pi_0 f(-, x')) \\ & \wedge \\ & x' \in t(\pi_1 f(x, -)) \end{aligned}$$

Note as an example, we have the beautiful equation showing Nash equilibria are not a primitive but rather a derived concept.

$$\text{Nash} = \text{argmax} \otimes \text{argmax}$$

See [11] for more details. This equation also highlights how our product of selection functions can be seen concretely as the two games played in parallel with the famous Nash equilibrium of Prisoners Dilemma as the specific instance of two argmaxes run in parallel. In developing approximate game theory, we ensure it is compositional in that approximation interacts well with compositional operators such as the Nash product.

3 Approximation and Selection Functions

Given selection functions form a compositional model of game theory we can ask two natural questions: i) can we account for approximate equilibria in selection functions; and ii) is approximation compositional in that — for example — if we approximate a Nash product of two games, do we get the Nash product of two approximations? We answer both questions positively — the second to our surprise as there is no particular reason to believe that approximation of a structured game preserves that structure. The fact it does suggests the compositional approach to game theory and the approach to approximation advocated here are well designed concepts capturing fundamental logical structure which other approaches have missed. We begin by defining approximation in selection functions via a function

$$T : (\varepsilon : \mathbb{R}) \rightarrow \text{Sel}(X, V) \rightarrow \text{Sel}(X, V)$$

by

$$x \in (T_\varepsilon G)k \quad \text{iff} \quad (\exists k' \sim_\varepsilon k) \ x \in Gk'$$

where we write $k' \sim_\varepsilon k$ for $d(k', k) \leq \varepsilon$ using the sup-metric on functions defined above. What is nice about this definition is that not only is it extremely simple and thus elegant, but it also formalises the intuition that an ε -optimal choice for a utility function k is an actual optimal choice, but for a utility function k' differing no more than ε from k . Getting this definition right, and getting the right notion of morphism between selection functions, were the fundamental keys required to achieving the smooth mathematical theory we have been able to develop in this paper. The first thing to do is to relate our notion of approximation to the traditional notion of approximation that one finds in the literature.

Lemma 3.1. *Let $\varepsilon \geq 0$. Then $T_\varepsilon(\text{argmax}) \subseteq 2\varepsilon\text{-argmax}$*

Proof. Let $x \in T_\varepsilon(\text{argmax})k$ for some utility function $k : X \rightarrow V$. Then there is a $k' \sim_\varepsilon k$ such that $x \in \text{argmax}(k')$, ie for all $x' \in X$, we have $k'x \geq k'x'$. But since $k' \sim_\varepsilon k$, we thus have $kx \geq kx' - 2\varepsilon$. Thus $x \in 2\varepsilon\text{-argmax } k$ □

Assuming X has a decidable equality, which includes the key case where X is finite, we can go further. Note the use of decidable equality is to ensure k' below is well defined.

Lemma 3.2. *Let $\varepsilon \geq 0$. If X has a decidable equality, then*

$$\varepsilon\text{-argmax} \subseteq T_\varepsilon(\text{argmax})$$

Proof. Let $x \in \varepsilon\text{-argmax}(k)$ for a utility function $k : X \rightarrow V$. Then for all $y \in X$, $kx \geq ky - \varepsilon$. Define $k' : X \rightarrow V$ by $k'x = kx + \varepsilon$ and $k'y = ky$ for all $y \neq x$. Clearly $k \sim_\varepsilon k'$. Further, for any $y \neq x$, we have $k'x \geq k'y$. Thus $x \in \text{argmax}(k')$ and so $x \in T_\varepsilon(\text{argmax})(k)$ $\square$

Moving to a categorical analysis, the first thing we would like to establish is the functoriality of T_ε . This is non-trivial as the needs of approximation put constraints on how lens maps back-propagate values. In essence, we must restrict to *short* lens maps which we define now

Definition 3.3. *A short lens map is a lens map $\alpha : (X, V) \rightarrow (X', V')$ where, for every $x \in X$, $\alpha(x, -) : V' \rightarrow V$ is a short map, ie non-expanding.*

Short maps are a natural condition — the condition reflects desire that cost (or backpropagation) doesn't inflate differences in the values to be backpropagated. Were we to allow such inflation, system behaviour could well become chaotic as small amounts of sub-optimality would be magnified into large system reconfiguration. Note the identity is a short map and the composite of short maps is short and hence the category of lenses restricts to a category of lenses and short maps denoted Lens_s . Similarly, we define the category of selection functions and maps whose underlying lens map is short by Sel_s . And, finally, the tensor product on lenses and selection functions restricts to a monoidal structure on Sel_s .

Lemma 3.4. *Let $\alpha : (X, V) \rightarrow (X', V')$ be a short lens map. If $k, k' : X' \rightarrow V'$ is such that $k \sim_\varepsilon k'$, then $k\alpha \sim_\varepsilon k'\alpha$*

Proof. Direct from the definition of what a short lens map is. $\square$

We can now prove functoriality:

Lemma 3.5. *Let $\alpha : s \rightarrow t$ be a map of selection functions over the short lens map $\alpha : (X, V) \rightarrow (X', V')$ of lenses. Then α also defines a map of selection functions $T_\varepsilon s \rightarrow T_\varepsilon t$*

Proof. Consider $x \in X$ and $k : X' \rightarrow V'$. We need to show if $\alpha x \in (T_\varepsilon t)k$, then $x \in (T_\varepsilon s)(k\alpha)$. The assumption implies there is a $k_1 : X' \rightarrow V'$ such that $k \sim_\varepsilon k_1$ and $\alpha x \in tk_1$. As $\alpha : s \rightarrow t$, this means $x \in s(k_1\alpha)$. By the previous lemma, $k\alpha \sim_\varepsilon k_1\alpha$ and so $x \in T_\varepsilon s(k\alpha)$ as required. This shows that any map $\alpha : s \rightarrow t$ induces a map $\alpha : T_\varepsilon s \rightarrow T_\varepsilon t$. Preservation of composition and identities is now trivial. $\square$

Next we consider the role/structure of the parameter ε in approximation. The following graded structure is what one expects in many quantitative situations.

Lemma 3.6. *The following are easily provable*

- $T_0 G = G$
- If $\varepsilon \leq \varepsilon'$, then $T_\varepsilon G \subseteq T_{\varepsilon'} G$
- $T_\varepsilon(T_\delta G) \subseteq T_{\varepsilon+\delta} G$

Proof. Straightforward $\square$

3.1 Compositional Approximation

In this subsection we discuss the relationship between the compositional structure of selection functions and approximation of selection functions. To do this we first prove a small lemma

Lemma 3.7. *Let $k, k' : X \times X' \rightarrow V \times V'$ be functions such that $k \sim_\varepsilon k'$. Then for every x and x' ,*

$$\pi_0 k(-, x') \sim_\varepsilon \pi_0 k'(-, x') \text{ and } \pi_1 k(x, -) \sim_\varepsilon \pi_1 k'(x, -)$$

Proof. Straightforward calculation □

Now, consider $G \otimes H$, the Nash product of selection functions G and H . What do its approximate equilibria look like? Is there any structure here or does it just consist of a fuzzy ball of extra equilibria enveloping G and H ? The following shows that structure is preserved, ie that an approximate equilibrium of a Nash product is infact an equilibrium of a Nash product.

Lemma 3.8. *Let $G : \text{Sel}(X, V)$ and $H : \text{Sel}(X', V')$ be selection functions. Then*

$$\mathsf{T}_\varepsilon(G \otimes H) \subseteq \mathsf{T}_\varepsilon G \otimes \mathsf{T}_\varepsilon H$$

Proof. Let $(x, x') \in X \times X'$ and $k : X \times X' \rightarrow V \times V'$. Further assume $(x, x') \in \mathsf{T}_\varepsilon(G \otimes H) k$. Then there is a $k' : X \times X' \rightarrow V \times V'$ such that $k \sim_\varepsilon k'$ and $(x, x') \in (G \otimes H) k'$. By definition of $G \otimes H$, this means

$$x \in G(\pi_0 k'(-, x')) \text{ and } x' \in H(\pi_1 k'(x, -))$$

By the previous lemma

$$x \in \mathsf{T}_\varepsilon(G)(\pi_0 k(-, x')) \text{ and } x' \in \mathsf{T}_\varepsilon(H)(\pi_1 k(x, -))$$

and hence $(x, x') \in (\mathsf{T}_\varepsilon G \otimes \mathsf{T}_\varepsilon H) k$. Thus $\mathsf{T}_\varepsilon(G \otimes H) \subseteq \mathsf{T}_\varepsilon G \otimes \mathsf{T}_\varepsilon H$ as required. □

The above lemma states that the approximation of a Nash product is itself a Nash product. Noting the contravariance in our notion of morphism of selection functions, we can summarise the above as:

Lemma 3.9. *The functor $\mathsf{T}_\varepsilon : \text{Sel}_s \rightarrow \text{Sel}_s$ is lax monoidal functor*

However, it would be nice if the reverse were true as then we could *find* and approximate equilibrium for a Nash equilibria by merely approximating the components. Not suprisingly, the ability to take this extra step depends on the existence of sufficient utility functions as we saw in lemma 3.2 which in turn rests on the decidability of equality on actions. The key lemma is the following which uses the decidable equality to ensure the function k' below is well defined.

Lemma 3.10. *Let X, X' have decidable equality. Further, let $k : X \times X' \rightarrow V \times V'$, $x \in X$ and $x' \in X'$. Even further, let $g \sim_\varepsilon \pi_0 k(-, x')$ and $h \sim_\varepsilon \pi_1 k(x, -)$. Then there is a $k' : X \times X' \rightarrow V \times V'$ such that*

- $k' \sim_\varepsilon k$
- $g = \pi_0 k'(-, x')$; and
- $h = \pi_1 k'(x, -)$.

Proof. Define $k' : X \times X' \rightarrow V \times V'$ as follows

$$\begin{aligned} k'(z, z') &= k(z, z') && \text{if } z \neq x, z' \neq x' \\ k'(x, z') &= (\pi_0 k(x, z'), h z') && \text{if } z' \neq x' \\ k'(z, x') &= (g z, \pi_1 k(z, x')) && \text{if } z \neq x \\ k'(x, x') &= (g x, h x') && \text{otw} \end{aligned}$$

The required properties of k' are now easy to verify. □

Now we can prove the reverse containment

Lemma 3.11. *Let X, X' have decidable equality. Let $G : \text{Sel}(X, V)$ and $H : \text{Sel}(X', V')$ be selection functions. Then*

$$\mathsf{T}_\varepsilon G \otimes \mathsf{T}_\varepsilon H \subseteq \mathsf{T}_\varepsilon (G \otimes H)$$

Proof. Let $(x, x') \in X \times X'$ and $k : X \times X' \rightarrow V \times V'$ be such that $(x, x') \in (\mathsf{T}_\varepsilon G \otimes \mathsf{T}_\varepsilon H)k$. Then

$$x \in (\mathsf{T}_\varepsilon G)(\pi_0 k(-, x')) \quad \text{and} \quad x' \in (\mathsf{T}_\varepsilon H)(\pi_1 k(x, -))$$

Thus there is a function $g : X \rightarrow V$ and one $h : X' \rightarrow V'$ such that

$$x \in Gg \quad \text{and} \quad x' \in Hh \quad \text{and} \quad g \sim_\varepsilon \pi_0 k(-, x') \quad \text{and} \quad h \sim_\varepsilon \pi_1 k(x, -)$$

By lemma 3.10, there is a map $k' : X \times X' \rightarrow V \times V'$ such that

$$x \in G(\pi_0 k'(-, x')) \quad \text{and} \quad x' \in H(\pi_1 k'(x, -)) \quad \text{and} \quad k' \sim_\varepsilon k$$

Thus $(x, x') \in (G \otimes H)k'$ and hence $(x, x') \in \mathsf{T}_\varepsilon (G \otimes H)k$ □

This is a very strong — and surprising result — that for action sets with decidable equality, approximation is not merely lax monoidal, but actually monoidal. This we did not expect and, indeed, when it comes to sequential composition of open games, we can only show one containment. Sequential composition of games leads us from selection functions to Open Games.

4 Approximation and Open Games

Selection functions form a simple model of compositional game theory via the Nash product/parallel composition of selection functions. However, what about the sequential composition of selection functions — after all parallel and sequential composition are intimately bound in monoidal category theory, string diagrams etc. Sequential composition of selection functions has been considered [5] but the utility function of the first game is modelled exogenously ... what happens in practice is that the utility function of the first game is actually computed endogenously. Open Games [9] address this problem perfectly. For the price of a little more algebraic structure, they form a compositional model of game theory which includes a number of operators for compositionally building large complex games from smaller and simpler games. Chief amongst these operators are sequential composition but also choice operators, and operators for iterated game theory [10]. This section generalises approximation to open games.

4.1 Open Games

The key concept which introduced open games is the following:

Definition 4.1 (Open Game). *Let X, Y, R and S be sets. An open game $G = (\Sigma_G, P_G, C_G, E_G) : (X, S) \rightarrow (Y, R)$ consists of:*

- a set Σ_G of strategy profiles,
- a play function $P_G : \Sigma_G \rightarrow (X \rightarrow Y)$,
- a coulity function $C_G : \Sigma_G \rightarrow (X \times R \rightarrow S)$, and
- an equilibrium function $E_G : X \times (Y \rightarrow R) \rightarrow \mathsf{P}(\Sigma_G)$.

Intuitively, the set X contains the states of the game, Y the moves, R the utilities and S the couilities. The set Σ_G contains the strategies we are trying to pick an optimal one from. The play function P_G selects a move given a strategy and a state, while the couility function C_G computes the couility extruded from the game, given a strategy, state and utility. Finally, if $\sigma \in E_G x k$, then σ is an optimal strategy in state x and with utility given by $k : Y \rightarrow R$. Just as lenses systematise the presentation of selection functions, they also systematise the presentation of open games as can be found in [11, 9].

Lemma 4.2. *An open game $G : (X, S) \rightarrow (Y, R)$ is given by a set Σ of strategies and, for each $\sigma \in \Sigma$: i) a lens $G_\sigma : (X, S) \rightarrow (Y, R)$; and ii) a predicate*

$$E_\sigma \subseteq X \times (Y \rightarrow R)$$

We now restrict all lenses occurring in open games to be short. This is important not only in functoriality as before, but also for the compositional treatment of approximation for sequential games. Notice the similarity of the lens predicate E_σ and a selection function. But notice also differences — a selection function is defined over a pair of sets, ie an object of Lens, while an open game is defined over a family of lens maps. There are other notions of open games, in particular Gavranovic Games [3] which deepen the relationship with lenses by defining open games to be parameterised maps in the category of lenses. We expect our techniques apply to Gavranovic games in the obvious way.

4.2 Approximation and Open Games

The methodology developed for selection functions clearly generalises to open games as follows:

Definition 4.3. *Let $G : (X, S) \rightarrow (Y, R)$ be an open game with strategy set Σ and equilibrium predicates E_σ for $\sigma \in \Sigma$. Further let $\varepsilon \in R$. Then define $T_\varepsilon G : (X, S) \rightarrow (Y, R)$ to be the open game with i) the same strategy set Σ ; ii) lens structure $(T_\varepsilon G)_\sigma = G_\sigma$; and iii) equilibrium given by*

$$x \in E_{T_\varepsilon G, \sigma} k \text{ iff } (\exists k' \sim_\varepsilon k) x \in E_{G, \sigma} k'$$

The similarity of approximation for open games with that for selection functions is obvious and highlights a strength of the methodology developed here — it is likely to generalise to many other situations as discussed in the conclusions and future work. This similarity also means that lemma 3.6 carries over to the setting of open games. Next we consider functoriality of approximation for open games

Definition 4.4. *The category of open games Op has open games as objects, and a morphism from the open game $(\Sigma, G, E) : (X, S) \rightarrow (Y, R)$ to the open game $(\Sigma', G', E') : (X', S') \rightarrow (Y', R')$ consists of a function $f : \Sigma \rightarrow \Sigma'$, short lens maps $\alpha : (X, S) \rightarrow (X', S')$ and $\beta : (Y, R) \rightarrow (Y', R')$ such that*

$$(\forall \sigma \in \Sigma)(\forall x \in X)(\forall k : Y' \rightarrow R') \alpha x \in E'_{f\sigma} k \text{ implies } x \in E_\sigma(k.\beta)$$

Notice how the contravariance in maps of selection functions is replicated here in maps of open games. This will be another hallmark of a general theory of approximation for systems consisting of utility indexed predicates. And again notice the use of the shortness of the lens maps α and β . The proof of the following lemma is the natural generalisation of the analogous one for selection functions.

Lemma 4.5. *For $\varepsilon \in R$, the assignment sending an open game G to an open game $T_\varepsilon G$ extends to a functor $T_\varepsilon : \text{Op} \rightarrow \text{Op}$*

The final part of the mathematical development of approximation for selection functions considered the monoidal product formalising Nash equilibria which showed that $T_\varepsilon(G \otimes H) \subseteq T_\varepsilon G \otimes T_\varepsilon H$, and that $\subseteq$ can be replaced by $=$ when moves are decidable. Again, this lemma generalises to open games — see [9] for exact the definition of the Nash product for open games.

Lemma 4.6. *The functor $\top_\varepsilon : \text{Op} \rightarrow \text{Op}$ is lax monoidal with respect to the Nash tensor on open games, and monoidal where the type of moves (inputs to utility) is decidable.*

Proof. The proof follows the same structure as Lemma 3.9. $\square$

One of the great advantages of open games is they compositionally model the sequential composition of games. Indeed, if we were to form a category whose objects are pairs of sets and whose morphisms are open games, the sequential composition of open games would be the composition in that category.

Definition 4.7. *Let $(\Sigma, G, E) : (X, S) \rightarrow (Y, R)$ be an open game and $(\Omega, H, B) : (Y, R) \rightarrow (Z, T)$ be another open game. Their sequential composition is the open game $H \circ G : (X, S) \rightarrow (Z, T)$ defined by*

- *Strategies of $H \circ G$ are $\Sigma \times \Omega$*
- *The lens $(H \circ G)(\sigma, \tau)$ is defined to be the lens composition $H\tau.G\sigma$*
- *The equilibrium predicate is defined by*

$$x \in E_{HG,(\sigma,\tau)}(k : Z \rightarrow T) \text{ iff } x \in E_{G,\sigma}(k.H\tau) \wedge G_\sigma x \in E_{H,\tau} k$$

Just like with the Nash product, we can ask ourselves the if the approximate equilibria of a sequential game has any structure, eg is it composed of equilibria for the sequential composition of two approximated games. Once more, the answer is yes!

Lemma 4.8. *Let $(\Sigma, G, E) : (X, S) \rightarrow (Y, R)$ and $(\Omega, H, B) : (Y, R) \rightarrow (Z, T)$ be open games. Then*

$$x \in E_{\top_\varepsilon(H \circ G),(\sigma,\tau)}(k : Z \rightarrow T) \text{ implies } x \in E_{\top_\varepsilon H \circ \top_\varepsilon G,(\sigma,\tau)}(k : Z \rightarrow T)$$

for any $x \in X$, $k : Z \rightarrow T$, $\sigma \in \Sigma$ and $\tau \in \Omega$

Proof. Assume $x \in X$, $k : Z \rightarrow T$, $\sigma \in \Sigma$ and $\tau \in \Omega$ and $x \in E_{\top_\varepsilon(H \circ G),(\sigma,\tau)}(k : Z \rightarrow T)$. By definition of $\top_\varepsilon$, we have a $k' : Z \rightarrow T$ such that $k \sim_\varepsilon k'$ and $x \in E_{H \circ G,(\sigma,\tau)}(k' : Z \rightarrow T)$. By definition, this means $x \in E_{G,\sigma}(k'.H\tau)$ and $G_\sigma x \in E_{H,\tau} k'$. Since $H\tau$ is short, we have $k.H\tau \sim_\varepsilon k'.H\tau$ and hence

$$x \in E_{\top_\varepsilon G,\sigma}(k.H\tau) \text{ and } G_\sigma x \in E_{\top_\varepsilon H,\tau} k$$

Thus $x \in E_{\top_\varepsilon H \circ \top_\varepsilon G,(\sigma,\tau)}(k : Z \rightarrow T)$ $\square$

Unlike with the Nash product, it is unlikely we can prove the reverse direction as, given $x \in E_{\top_\varepsilon G,\sigma}(k.H\tau)$, all we could conclude is there is a $k' \sim_\varepsilon k.H\tau$ such that $x \in E_{G,\sigma} k'$ but vital and needed structure is lost.

5 Metric Game Theory

Apart from solving the concrete problem of when is a move almost optimal in a selection function or open game, our Approximate Game Theory permits further theoretical developments. The one we sketch here we call *Metric Game Theory* which seeks to ask the question of how far apart are two games. This is important, eg in a compositional model we might want to know if we replace sub-games/components of one game with other sub-game/components, can we quantitatively measure the effect. In more detail, we might ask questions such as if $G \sim_\varepsilon G'$, what can we say about the distance between $G \otimes H$ and $G' \otimes H$. In this section we develop the basics of Metric Game Theory. We focus on selection functions as the theory seems to generalise to open games. We also only measure distances between selection functions over the same lens — we expect we can use the fibrational structure of selection functions over lenses to enable the computation of distances between selection functions over different lenses. But we leave a proper treatment of *Metric Game Theory* to a full paper as the subject clearly warrants.

Definition 5.1. Let $s, s' \in \text{Sel}(X, V)$ be selection functions. We define

$$d(s, s') = (\inf \varepsilon : \mathbb{R}) \ (s \subseteq T_\varepsilon s' \ \wedge \ s' \subseteq T_\varepsilon s)$$

In the above, we are essentially using the Hausdorff distance on subsets. As a result, the above construction gives a pseudometric, ie there is no requirement that $d(x, y) = 0$ implies $x = y$. We list two obvious properties of the pseudometric on $\text{Sel}(X, V)$ showing how distances between games are preserved by compositional structure.

Lemma 5.2. Assuming a decidable equality on X, X' , the following are true

- Let s be a selection function. Then $d(s, T_\varepsilon s) \leq \varepsilon$
- Let $s, s' \in \text{Sel}(X, V)$ and $t, t' \in \text{Sel}(X', V')$. If $d(s, s') \leq \varepsilon$ and $d(t, t') \leq \varepsilon$, then $d(s \otimes t, s' \otimes t') \leq \varepsilon$

Proof. The first property follows from the graded structure of T_ε . For the second, note $s \subseteq T_\varepsilon s'$ and $t \subseteq T_\varepsilon t'$. Thus $s \otimes t \subseteq T_\varepsilon s' \otimes T_\varepsilon t' = T_\varepsilon(s' \otimes t')$ $\square$

Finally, note this states if we replace a subcomponent of a Nash product by one ε -different, the overall Nash product is no further than ε -apart. This is an ideal property to have in a quantitative compositional theory of games. The analogous property for the Nash product of open games also holds. However, note that for sequential composition of open games we only have $T_\varepsilon(H \circ G) \subseteq T_\varepsilon H \circ T_\varepsilon G$. Thus, replacing subgames with other subgames can increase distance between the overall games but (not shown here) in a bounded way — enough to give a theory of *uncertainty budgets* for approximation of open games.

6 Conclusions and Future Work

In summary, we have tackled the important problem of approximation for game theory. This is important as exact equilibria are often either impossible to determine due to inherent uncertainty in underlying data, or undesirable because the cost of exact computation is too high in comparison to approximate computation. In developing our approach to approximate game theory, it was vital to ensure our theory of approximation was compositional in that it respected the compositionality of game theory. Our fundamental idea was that a strategy was approximately optimal for a given utility function if it is optimal for an approximation of that utility function. This idea was developed for two models of game theory - firstly selection functions and secondly open games. In both, approximation was shown to preserve compositional structure. Finally, we showed how approximation and compositionality can be used to develop *Metric Game Theory* which allows one to replace approximately equal subcomponents of a Nash product and produce guarantees on how close the overall resulting systems are.

By opening up a new field, this paper has much that can be done to expand it. Firstly, there are a number of other operators in open games which the methods developed here could be applied to. These include choice, contextualisation and infinite iteration [10]. Secondly, these methods could also be extended to more advanced forms of open games, eg i) probabilistic open games using distance metrics such as Kantorovich on probability distributions; or ii) games based upon optics [17]. Thirdly, the generality of our ideas means applications to other forms of games are natural. For example, in Markov Decision Processes [6], one can consider coalgebras

$$\langle u, t \rangle : X \rightarrow R \times (A \rightarrow DX)$$

where X is a state space, R is a type of utility, A is a set of actions, D is the probability monad, $u : X \rightarrow R$ is a utility function and $t : X \times A \rightarrow DX$ is a probabilistic transition function. Within MDPs one defines optimal policies which we can denote Opt . Our methodology suggests defining an approximate version

$$\pi \in \text{Opt } T_\varepsilon \langle u, t \rangle \text{ iff } \pi \in \text{Opt } \langle u', t \rangle$$

where $u' \sim_\varepsilon u$. One could even be more general and consider tweaks to the structure of t ! Most excitingly, machine learning has features in common with game theory in that the former finds a parameter to minimise a loss function while the later finds a strategy to maximise a utility function [3, 4]. Thus it is possible that the ideas developed here could align with learning based upon back propagation [7] and, in particular, categorical deep learning [8] and its special case of geometric deep learning. This would contribute to the huge problem of XAI. It would be good to see what an implementation in code of our ideas looks like, eg in the implementation of open games [12]. The breadth of application areas also suggests more theoretical work unifying these disparate examples, eg via a theory of approximation for utility indexed predicates, or maybe predicates fibred over metric spaces.

References

- [1] Michael Abbot, Thorsten Altenkirch & Neil Ghani (2005): *Containers: Constructing strictly positive types*. *Theoretical Computer Science* 342 (1), 3–27, doi:10.1016/j.tcs.2005.06.002.
- [2] Flavien Breuvar, Dylan McDermott & Tarmo Uustalu (2023): *Canonical Gradings of Monads*. *Electronic Proceedings in Theoretical Computer Science* 380, p. 1–21, doi:10.4204/EPTCS.380.1.
- [3] Matteo Capucci, Bruno Gavranović, Jules Hedges & Eigil Fjeldgren Rischel (2022): *Towards Foundations of Categorical Cybernetics*. *Electronic Proceedings in Theoretical Computer Science* 372, p. 235–248, doi:10.4204/eptcs.372.17.
- [4] Geoff S. H. Crutwell, Bruno Gavranovic, Neil Ghani, Paul W. Wilson & Fabio Zanasi (2021): *Categorical Foundations of Gradient-Based Learning*. CoRR abs/2103.01931, doi:10.48550/arXiv.2103.01931. Available at <https://arxiv.org/abs/2103.01931>.
- [5] Martín Escardó & Paulo Oliva (2010): *Sequential games and optimal strategies*. *Proceedings of the Royal Society A* 467, pp. 1519–1545, doi:10.1098/rspa.2010.0471.
- [6] Frank Feys, Helle Hvid Hansen & Lawrence S. Moss (2018): *Long-Term Values in Markov Decision Processes, (Co)Algebraically*. In: *14th International Workshop on Coalgebraic Methods in Computer Science (CMCS), Coalgebraic Methods in Computer Science LNCS-11202*, pp. 78–99, doi:10.1007/978-3-030-00389-0_6.
- [7] Brendan Fong, David I. Spivak & Rémy Tuyéras (2019): *Backprop as Functor: A compositional perspective on supervised learning*. In: *LICS’19, IEEE*, pp. 1–13, doi:10.1109/LICS.2019.8785665.
- [8] Bruno Gavranović, Paul Lessard, Andrew Dudzik, Tamara von Glehn, João G. M. Araújo & Petar Veličković (2024): *Categorical Deep Learning: An Algebraic Theory of Architectures*. arXiv:2402.15332.
- [9] Neil Ghani, Jules Hedges, Viktor Winschel & Philipp Zahn (2018): *Compositional Game Theory*. In: *LICS’18, ACM*, pp. 472–481, doi:10.1145/3209108.3209165.
- [10] Neil Ghani, Clemens Kupke, Alasdair Lambert & Fredrik Nordvall Forsberg (2018): *A compositional treatment of iterated open games*. *Theor. Comput. Sci.* 741, pp. 48–57, doi:10.1016/j.tcs.2018.05.026.
- [11] Jules Hedges (2016): *Towards compositional game theory*. Ph.D. thesis, Queen Mary University of London.
- [12] Jules Hedges, Philipp Zahn, André Videla & Sjoerd Visscher: *Open Games in Haskell*. Available at <https://github.com/jules-hedges/open-games-hs>.
- [13] Bart Jacobs (2016): *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, doi:10.1017/CB09781316823187.

- [14] Kevin Leyton-Brown & Yoav Shoham (2008): *Essentials of Game Theory: A Concise Multidisciplinary Introduction*. Morgan & Claypool, doi:10.1007/978-3-031-01545-8.
- [15] Saunders MacLane (1971): *Categories for the working mathematician*. Springer-Verlag. Graduate Texts in Mathematics, Vol. 5.
- [16] Martin Osbourne & Ariel Rubinstein (1994): *A course in game theory*. MIT Press.
- [17] Mitchell Riley (2018): *Categories of Optics*, doi:10.48550/arXiv.1809.00738. arXiv:1809.00738.
- [18] Joel Watson (2002): *Strategy: an introduction to game theory*. vol 139, WW Norton New York.

Fibrational Perspectives on Determinization of Finite-State Automata

Thea Li

Université Paris Cité
Paris, France

thea.li@etu.u-paris.fr

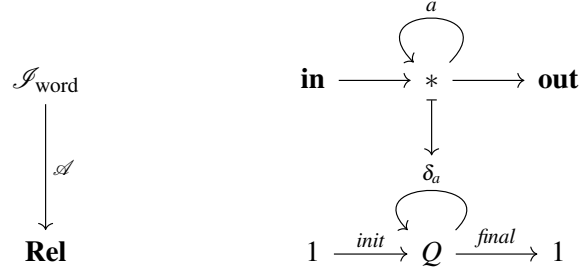
Colcombet and Petrişan argued that automata may be usefully considered from a functorial perspective, introducing a general notion of “ $\mathcal{V}$ -automaton” based on functors into $\mathcal{V}$. This enables them to recover different standard notions of automata by choosing $\mathcal{V}$ appropriately, and they further analyzed the determinization for **Rel**-automata using the Kleisli adjunction between **Set** and **Rel**. In this paper, we revisit Colcombet and Petrişan’s analysis from a fibrational perspective, building on Melliès and Zeilberger’s recent alternative but related definition of categorical automata as functors $p : \mathcal{Q} \rightarrow \mathcal{C}$ satisfying the finitary fiber and unique lifting of factorizations property. In doing so, we improve the understanding of determinization in three regards: Firstly, we carefully describe the universal property of determinization in terms of forward-backward simulations. Secondly, we generalize the determinization procedure for **Rel** automata using a local adjunction between **Span(Set)** and **Rel**, which provides us with a canonical forward simulation. Finally we also propose an alternative determinization based on the multiset relative adjunction which retains paths, and we leverage this to provide a canonical forward-backward simulation.

1 Introduction

One motivation for studying automata from a categorical point of view is that it enables us to reason about properties of automata abstractly. In particular, this is useful when considering constructions where canonicity might be desirable, such as for determinization of non-deterministic automata. Commonly one considers automata as algebras or coalgebras, see among others [1, 6, 10, 3].

A different approach is taken by Colcombet and Petrişan [5]. They introduced a general notion of $\mathcal{V}$ -automaton based on functors into $\mathcal{V}$, which enabled them to recover standard notions of automata by instantiating $\mathcal{V}$ appropriately. In particular, they show that non-deterministic automata can be seen as **Rel**-automata, and they further provide a determinization for **Rel**-automata using the Kleisli-adjunction between **Set** and **Rel**. The **Rel**-automata approach to non-deterministic automata is practical for studying logical aspects of automata, yet it does not retain some of the computational content. Consequently, we revisit Colcombet and Petrişan analysis of determinization from the point of view of Melliès and Zeilberger’s [7] alternative approach, using functors with the *unique lifting of factorizations* (ULF) property, that ameliorates the loss of computational content and generalizes **Rel**-automata. Their approach describes non-deterministic automata as ULF-functors over a category of labels, with finite fibers.

Both [7] and [5] provide a definition of deterministic automata, which turn out to be equivalent under the Grothendieck construction. As mentioned, determinization categorically has previously been studied by Colcombet and Petrişan for **Rel**-automata, but is yet to be done Melliès and Zeilberger’s approach. In this work, we detail the determinization procedure for **Rel** and provide a generalization of this to the setting of [7], recovering a similar universal property. We then discuss the links between our construction and the usual algorithmic procedure first presented by Rabin and Scott [9]. As this procedure identifies

Figure 1: a non-deterministic automata as a functor into **Rel**

paths with the same label, which is not always desirable, we construct an alternative determinization using multisets that is path-relevant.

1.1 Outline

We recall in Section 2 the two definitions of non-deterministic automata as functors and the common definition of a deterministic automaton. In Section 3 we revisit the determinization procedure for **Rel**-automata and in Section 4 we construct a generalization of this procedure to the **Span(Set)** case. We then construct an alternative determinization in Section 5 with a stronger universal property, though the downside of this one is a significant increase in size.

2 Preliminaries

2.1 Automata as functors

In [5], Colcombet and Petrişan consider functors as generalized automata. They view the domain as the category of input words and the codomain as the category of outputs, and the functor itself as specifying the behavior of the automaton. Additionally, to specify the accepted input, they require a full subcategory of the category of inputs, that in practice amounts to choosing initial and final states.

Definition 2.1. [5, Def. 2.1] A $\mathcal{V}$ -automaton is a functor $\mathcal{A} : \mathcal{I} \rightarrow \mathcal{V}$, together with a full subcategory inclusion $i : \mathcal{O} \rightarrow \mathcal{I}$

This definition has the advantage to be very simple categorically and easy to manipulate. Moreover, the accepted language can be defined naturally as the composition of the automata and its full subcategory inclusion.

Definition 2.2. [5, Def. 2.1] The language accepted by a $\mathcal{V}$ -automaton $\mathcal{A} : \mathcal{I} \rightarrow \mathcal{V}$ is the composition $\mathcal{A} \circ i : \mathcal{O} \rightarrow \mathcal{V}$.

To model classical non-deterministic automata without epsilon transitions they particularize the definition to the case where $\mathcal{V}$ is **Rel**, the category of sets and relations. Given a fixed automaton over the alphabet Σ , they take $\mathcal{I}$ to be the one object category associated to the free monoid Σ^* , to which they add two objects "in" and "out" and two morphisms that act as begin- and end-input markers. $\mathcal{O}$ is then the full subcategory generated by in and out, we denote this by $\mathcal{I}_{\text{word}}$. The functor maps the base object of the free monoid to the set of states, each letter to the relation generated by the transition function, "in"

and "out" are both mapped to 1 and the begin- and end-input markers to relations pointing out initial and final states, see Fig. 1.

In the remainder of the paper, we will call automata with codomain **Rel**, **Rel**-automata, regardless of domain.

2.2 Non-deterministic automata as ULF functors

The **Rel**-automata approach to non-deterministic automata is language oriented and models very well the logical aspects of automata. Yet, as the functor is mapping into **Rel**, it only interprets transitions as ways of relating states under a label and does not account for which states it passed along the way, nor does it account for the different ways of doing so. It thus equates different paths, and we lose some of the computational content of automata.

Mellies and Zeilberger [7] introduced a categorification of non-deterministic automata (without epsilon transitions) using ULF-functors, that is closer to the classical definition, ameliorates the loss of computational content, and is actually a generalization of **Rel**-automata. The idea is for the domain to be the underlying directed graph of the automaton and the codomain to be the alphabet, with the ULF functor specifying the label of a path. The ULF-property then describes an important feature of automata, that a path α labelled by a word w can be factored in subpaths along the letters of w .

Definition 2.3 (ULF-functor). A functor p is ULF if for each factorization of a morphism $p(\alpha) = uv$ there are unique morphisms β and γ such that $\alpha = \beta\gamma$ and $p(\beta) = u$ and $p(\gamma) = v$.

Definition 2.4 (ULF-automaton [7, Def. 3.3]). An automaton over a category $\mathcal{C}$ is a tuple $(\mathcal{C}, \mathcal{Q}, p : \mathcal{Q} \rightarrow \mathcal{C}, q_0, Q_f)$ where p is a ULF functor with finite fibers, q_0 is an object of $\mathcal{Q}$, and Q_f is a set of objects in $\mathcal{Q}$.

The objects of $\mathcal{Q}$ correspond to states, q_0 to the initial state, and Q_f to the final states, while unfactorizable morphisms in $\mathcal{C}$ correspond to letters of our alphabet, and consequently arbitrary morphism to words. This naturally leads us to consider the following as its language.

Definition 2.5. [7, Def. 3.3] The language recognized by a ULF-automaton M with labelling functor p , is $L_M := \{w \mid \exists q_f \in Q_f, \exists \alpha : q_0 \rightarrow q_f, p(\alpha) = w\}$.

It turns out that ULF-automata generalizes **Rel**-automata, indeed any functor $\mathcal{I} \rightarrow \mathbf{Rel}$ can be extended to a functor $\mathcal{I} \rightarrow \mathbf{Span}(\mathbf{Set})$. Yet by a Grothendieck-like correspondence any pseudofunctor into **Span**(**Set**) corresponds to a ULF-functor [7], hence any **Rel**-automaton induces a ULF-automaton over $\mathcal{I}$. We can in fact characterize these as a particular type of ULF-functors.

Proposition 2.6. A functor $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ factors through **Rel** if and only if its corresponding functor lying over $\mathcal{C}$ (by the Grothendieck construction) is ULF and faithful.

Corollary 2.7. **Rel**-automata $\mathcal{C} \rightarrow \mathbf{Rel}$ factoring through **FinRel** are equivalent to faithful ULF-automata over $\mathcal{C}$.

We can, as noted in [7], represent ULF-automata as pseudofunctors $\mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ factoring through **Span**(**Fin**), up to choice of initial and final states. In practice, we will often directly work with the equivalent **Span**(**Set**)-functor rather than with the ULF-automaton, as it makes it easier to relate automata over the same category. We will henceforth refer to them as **Span**(**Set**)-automata.

We remark that we on occasion drop the condition that our **Span**(**Set**)-automata factor through **Span**(**Fin**) as it is not always important. In Section 4 all constructions preserve finiteness, and consequently the construction naturally restricts to **Span**(**Fin**), thus we will consider general **Span**(**Set**)-automata. However, in Section 5 whether constructions are finite is important and we adapt our definitions accordingly.

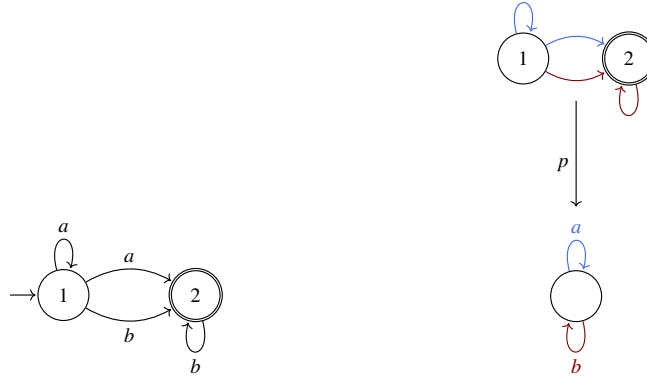


Figure 2: A non-deterministic automaton and its ULF-automaton interpretation (where color of the arrow indicates functorial assignment)

2.3 Deterministic Automata

Both [5] and [7] give definitions of a deterministic automaton. These definitions turn out to be equivalent under the Grothendieck construction given the right instantiations, and being able to reason using both approaches is useful.

By definition an automaton is deterministic when its transition relation is a functional relation, consequently, [5] models deterministic automata as **Rel**-automata that factor through **Set**.

Definition 2.8 ([5]). A deterministic automaton is a functor $\mathcal{I}_{\text{word}} \rightarrow \mathbf{Set}$, which maps *in* to 1 and *out* to 2.

For a ULF-functor to model a deterministic automaton it not only needs a unique lift of factorizations of words, as noted in [8], it requires every word to have a unique path given a starting state, meaning that any word $w : p(q) \rightarrow c$ has a unique lift $\alpha : q \rightarrow q'$.

Definition 2.9 (Discrete opfibration). A functor $p : \mathcal{Q} \rightarrow \mathcal{C}$ is a discrete opfibration if any morphism $w : p(q) \rightarrow c$ has a unique lift, i.e. there exists a unique $\alpha : q \rightarrow q'$ such that $p(\alpha) = w$.

Definition 2.10. [8, Def. 2.9] A deterministic automaton over a category $\mathcal{C}$ is a tuple $(\mathcal{C}, \mathcal{Q}, p : \mathcal{Q} \rightarrow \mathcal{C}, q_0, Q_f)$ where p is a discrete opfibration with finite fibers, q_0 is an object of $\mathcal{Q}$, and Q_f is a set of objects in $\mathcal{Q}$.

It turns out that these two definitions are equivalent given the right instantiations.

Proposition 2.11. By the Grothendieck construction, functors $\mathcal{C} \rightarrow \mathbf{Set}$ are equivalent to discrete opfibrations over $\mathcal{C}$.

Corollary 2.12. *Set*-automata $\mathcal{I}_{\text{word}} \rightarrow \mathbf{Set}$ are equivalent to deterministic automata over $\mathcal{I}_{\text{word}}$ where the fiber over *in* has one element and the fiber over *out* has two.

3 Determinisation for Rel

In order to determinize **Span**(**Set**)-automata, we first focus on the particular case of **Rel**-automata, previously studied in [5], this procedure will constitute in one building stone for our **Span**(**Set**)-automata

determinization. We recall the determinization procedure for **Rel**-automata in Section 3.1, and transpose it to faithful ULF-automata. Finally we discuss the difficulties of extending the construction to **Span(Set)**.

3.1 Determinisation for Rel

Determinizing a **Rel**-automaton $\mathcal{J} \rightarrow \mathbf{Rel}$ consists in transforming it into a functor $\mathcal{J} \rightarrow \mathbf{Set}$; to obtain such, one can postcompose by a functor $\mathbf{Rel} \rightarrow \mathbf{Set}$. As explained in [5], such a functor naturally arises, as **Rel** is the Kleisli category of the powerset monad on **Set**. Consequently, there is an adjunction $L : \mathbf{Set} \rightleftarrows \mathbf{Rel} : R$. The functor R sends objects A in **Rel** to their powersets $\mathcal{P}(A)$ and relations $r \subseteq A \times B$ to the functions $\mathcal{P}(A) \rightarrow \mathcal{P}(B)$ that sends a subset $S \subseteq A$ to the subset of all elements in B related to some element in S .

Construction 3.1. [5, Sec 3.3] Given a **Rel**-automaton $F : \mathcal{J} \rightarrow \mathbf{Rel}$, its determinization is given by

$$\text{Det}(F) := \mathcal{J} \xrightarrow{F} \mathbf{Rel} \xrightarrow{R} \mathbf{Set}$$

To be an appropriate determinization procedure the deterministic automaton should simulate the original automaton and recognize the same language. Simulations can be represented categorically as the following, which is inspired by simulations of labelled transition systems presented in [11] and a slight modification to the morphisms of automata presented in [5]. Moreover, this gives a suitable notion of morphism of automata.

Definition 3.2. Given two automata $F, F' : \mathcal{J} \rightarrow \mathbf{Rel}$, a (forward-backward) *simulation* from F to F' is a natural transformation $\alpha : F' \Rightarrow F$.

Indeed for any $c, d \in \text{ob}(\mathcal{J})$ and any word $w : c \rightarrow d$ if $(q', s) \in F(w) \circ \alpha_c$ then there is a state $q \in F'(c)$ such that $(q, q') \in F(w)$ and $(q, s) \in \alpha_c$, by naturality we also have that $(q', s) \in \alpha_d \circ F'(w)$ consequently there is a state $s' \in F(d)$ and $(s, s') \in F'(w)$ and $(q', s') \in \alpha_d$, that is, F' simulates F . We can make the reversed argument for any $(q', s) \in \alpha_d \circ F'(w)$, that is, we have a backward simulation.

Definition 3.3. **Rel**-automata with domain $\mathcal{J}$ and simulations form a category $\text{Aut}(\mathcal{J})$.

We can then justify the canonicity of the determinization by the existence of a canonical (*forward-backwards*) simulation from an automaton to its determinization, in which the initial state is simulated by an initial state and the final states are simulated by final states.

Proposition 3.4. Let $F : \mathcal{J} \rightarrow \mathbf{Rel}$ be an automaton, then there is a canonical (forward-backward) simulation from F to $\text{Det}(F)$ given by the counit of the Kleisli-adjunction between **Rel** and **Set**.

$$\begin{array}{ccc} \mathcal{J} & \xrightarrow{F} & \mathbf{Rel} \\ & & \searrow R \\ & & \mathbf{Set} \end{array} \quad \begin{array}{c} \xrightarrow{\quad} \\ \uparrow \varepsilon \\ \xrightarrow{\quad} \end{array} \quad \begin{array}{ccc} & \mathbf{Rel} & \\ & \xrightarrow{\quad} & \mathbf{Rel} \\ & \nearrow i & \end{array}$$

Having a canonical simulation is not enough to justify that the determinization preserves the language, as the simulation is one-sided, and as we have yet to address how initial and final states are determined for $\text{Det}(F)$. Under Grothendieck construction $\text{Det}(F)$ is

$$\text{Det}(M) := \left(\int RF, \mathcal{J}, \Pi : \int RF \rightarrow \mathcal{J}, q'_0, Q'_f \right)$$

We take as initial state $q'_0 = (p^{-1}(c), \{q_0\})$, where p is the ULF-functor corresponding to F , q_0 is the initial state of F and $c = p(q_0)$. The set of final states Q'_f are all states $(p^{-1}(q_f), S)$ such that q_f is a final state in F and $q_f \in S$. Given this it is straightforward to check that $\text{Det}(F)$ recognizes the same language as F .

3.2 Universal Property

Now that we have justified the canonicity of our determinization procedure, we would like to be able to relate it to other determinizations and other deterministic automata, that is, equip it with a universal property. To characterize the universal property we make use of the notion of bisimulation. For this purpose, recall that **Rel** is a dagger category taking $(-)^{\dagger}$ as the converse relation.

Definition 3.5. If $\alpha : F' \Rightarrow F$ is a natural transformation such that $(\alpha)^{\dagger}$ also is a natural transformation, then α is called a *bisimulation*.

Indeed, requiring $(\alpha)^{\dagger}$ to also be a simulation gives us the converse simulation, i.e. " F simulates F' ", by the simulation α .

As our canonical simulation is built out of a counit we make use of the triangle identity for unit and counit of adjoint functors to obtain the following universal property.

Proposition 3.6. Let $G : \mathcal{J} \rightarrow \mathbf{Set}$ be some deterministic automaton then any forward-backward simulation from F to G factors uniquely through the universal simulation to $\mathbf{Det}(F)$ and a bisimulation between $\mathbf{Det}(F)$ and G .

Proof. By one of the triangle identities for adjoint functors we have the following equality.

$$\begin{array}{ccc} \mathcal{J} & \xrightarrow{F} & \mathbf{Rel} \\ & \searrow G & \uparrow \alpha \\ & \mathbf{Set} & \nearrow i \end{array} = \begin{array}{ccccc} \mathcal{J} & \xrightarrow{F} & \mathbf{Rel} & \xlongequal{\quad} & \mathbf{Rel} \\ & \searrow G & \uparrow \alpha & \nearrow i & \uparrow \eta \\ & \mathbf{Set} & \xlongequal{\quad} & \mathbf{Set} & \nearrow \epsilon \\ & & & & \nearrow i \end{array}$$

The simulation between $\mathbf{Det}(F)$ and G is a bisimulation as any natural transformation generated by a natural transformation between functors into **Set** and an inclusion into **Rel**. $\square$

3.3 Difficulties in generalization to $\mathbf{Span}(\mathbf{Set})$ -automata

This determinization procedure has the advantage to be direct and fairly simple, but does not scale easily to **Span(Set)** as it is less well behaved than **Rel**. First, we do not have an adjunction between **Span(Set)** and **Set**, which was essential in the **Rel**-case for determinization and to establish the universal property. Second, as morphisms in **Span(Set)** are composed using pullbacks, functors out of or to **Span(Set)** tend to be lax or pseudo as pullbacks are only unique up to isomorphism.

4 Powerset determinization for $\mathbf{Span}(\mathbf{Set})$ -automata

In this section, we construct local adjunction between **Span(Set)** and **Rel** seen as 2-categories, and we show that this is sufficient to recover a determinization procedure and universal property analogous to the **Rel**-determinization for ULF-automata/**Span(Set)**-automata. We then discuss the link between the categorical determinization procedure and the algorithmic one.

4.1 Determinization procedure

As ULF-automata over $\mathcal{C}$ are equivalent to pseudofunctors $\mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ and deterministic ULF-automata are equivalent to functors $\mathcal{C} \rightarrow \mathbf{Set}$, it suffices to be able to build a functor $\mathcal{C} \rightarrow \mathbf{Set}$ out of a pseudo-functor $\mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ to determinize ULF-automata. As we know how to determinize

Rel-automata, it is sufficient for us to be able to turn a pseudo-functor $\mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ into a functor $\mathcal{C} \rightarrow \mathbf{Rel}$ in a canonical way. That is, it suffices to construct a lax functor $\mathbf{Span}(\mathbf{Set}) \rightarrow \mathbf{Rel}$ that strictifies the pseudo-functor to a strict functor $\mathcal{C} \rightarrow \mathbf{Rel}$ by postcomposition. To do so, we must work in a 2-categorical setting as we deal with pseudo-functors, and view $\mathbf{Span}(\mathbf{Set})$ and $\mathbf{Rel}$ as a 2-categorical construction.

Definition 4.1. $\mathbf{Span}(\mathbf{Set})$ is a 2-category with 2-morphisms given by morphisms of spans

$A \xleftarrow{f} S \xrightarrow{g} B$ to $A \xleftarrow{f'} S' \xrightarrow{g'} B$ which are functions $h : S \rightarrow S'$ such that:

$$\begin{array}{ccccc} & & S & & \\ & f \swarrow & \downarrow h & \searrow g & \\ A & \xleftarrow{f'} & S' & \xrightarrow{g'} & B \end{array}$$

In this context, we can view $\mathbf{Rel}$ as a 2-category in a similar way, 2-morphisms would be given by inclusions of relations, which indeed is just a morphism of spans.

There is a "forgetful" lax functor from $\mathbf{Span}(\mathbf{Set})$ to $\mathbf{Rel}$ that is identity on objects and maps spans to the relation generated by the image of the legs, we call this functor Im for image. As this functor identifies spans that generate the same relation, by post-composing it to a pseudo-functor such as our automaton F we obtain a functor into $\mathbf{Rel}$ that is strict. Post-composing an automaton $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ with $R \circ \text{Im}$ does indeed give us a determinization of F that preserves the language, as the image functor only identifies all paths with the same label between two given states.

Construction 4.2. Given a $\mathbf{Span}(\mathbf{Set})$ -automaton $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$, its determinization is given by

$$\text{Det}(F) := \mathcal{C} \xrightarrow{F} \mathbf{Span}(\mathbf{Set}) \xrightarrow{\text{Im}} \mathbf{Rel} \xrightarrow{R} \mathbf{Set}$$

As for $\mathbf{Rel}$ -automata, for our determinization to be appropriate, the deterministic automaton must simulate the original automaton and recognize the same language. This requires us to adapt our notion of simulation to the 2-categorical setting and to $\mathbf{Span}(\mathbf{Set})$.

Definition 4.3. Given two automata $F, F' : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$, a (*forward*) *simulation* from F to F' is a lax natural transformation $\alpha : F' \Rightarrow F$.

In relaxing our notion of simulation to a *lax* natural transformation we have a simulation by a similar reasoning as for $\mathbf{Rel}$ -automata by invoking the laxness, but not a backward simulation, as the lax naturality only guarantee a span-morphism in one direction. If such a simulation is pseudo we do obtain the backward simulation as well.

Definition 4.4. Given two automata $F, F' : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$, a (*forward-backward*) *simulation* from F to F' is a pseudo natural transformation $\alpha : F' \Rightarrow F$.

Definition 4.5. $\mathbf{Span}(\mathbf{Set})$ -automata with domain $\mathcal{C}$ and forward simulations form a category $\text{Aut}(\mathcal{C})$.

Though the image functor does not have any strict adjoints, it is the left adjoint of a local adjunction between $\mathbf{Rel}$ and $\mathbf{Span}(\mathbf{Set})$ seen as 2-categories, with the right adjoint being the inclusion. Using this local adjunction properly will enable us to prove the canonicity of our determinization procedure, as well as recover a universal property similar to one of $\mathbf{Rel}$. We start by recalling the definition and terminology used for local adjunctions.

Definition 4.6. Let $\mathcal{C}$ and $\mathcal{D}$ be bicategories, a local adjunction between $\mathcal{C}$ and $\mathcal{D}$ is comprised of two functors $L : \mathcal{C} \rightarrow \mathcal{D}$ and $R : \mathcal{D} \rightarrow \mathcal{C}$, inducing a family of adjunctions

$$\mathcal{D}(LA, B) \begin{array}{c} \xrightarrow{\quad} \\ \perp \\ \xleftarrow{\quad} \end{array} \mathcal{C}(A, RB)$$

natural in A and B . We use η and ε to denote units and counits on the local level.

To construct a canonical simulation we begin by constructing a canonical simulation from a **Span(Set)**-automaton and its **Rel**-ification, for this purpose we will make use of the following definition.

Definition 4.7. We denote by " η " : $i\text{Im} \Rightarrow \text{Id}_{\text{Span}(\text{Set})}$ the lax natural transformation that is the identity on each component with lax naturality given by the η the unit of the local adjunction $\text{Im} : \text{Span}(\text{Set})(A, B) \rightleftarrows \text{Rel}(A, B) : i$. We respectively denote by " ε " : $\text{Id}_{\text{Rel}} \Rightarrow \text{Im}i$ the lax natural transformation associated to the counit. Let c and d be sets and w and R be a span and a relation from c to d , respectively, then the lax naturality squares are the following for " η " and " ε ":

$$\begin{array}{ccc} c & \xrightarrow{i(\text{Im}(w))} & d \\ \text{Id}_c \downarrow & \nearrow \eta & \downarrow \text{Id}_d \\ c & \xrightarrow{w} & d \end{array} \qquad \begin{array}{ccc} c & \xrightarrow{R} & d \\ \text{Id}_c \downarrow & \nearrow \varepsilon & \downarrow \text{Id}_d \\ c & \xrightarrow{\text{Im}(i(R))} & d \end{array}$$

Lemma 4.8. Let $F : \mathcal{C} \rightarrow \text{Span}(\text{Set})$ be an automaton, then there is a canonical (forward) simulation from F to $i\text{Im}F$ given by the lax natural transformation " η " defined above.

$$\begin{array}{ccccc} \mathcal{C} & \xrightarrow{F} & \text{Span}(\text{Set}) & \xlongequal{\quad} & \text{Span}(\text{Set}) \\ & & \searrow \text{Im} & \Uparrow \eta & \nearrow i \\ & & \text{Rel} & & \end{array}$$

Using this we can now obtain a canonical simulation from an automaton to its determinization similar to the one for **Rel**.

Proposition 4.9. Let $F : \mathcal{C} \rightarrow \text{Span}(\text{Set})$ be an automaton, then there is a canonical (forward) simulation from F to $\text{Det}(F)$ given by pasting the (forward) simulation 4.8 and the counit of the Kleisli-adjunction between **Rel** and **Set**.

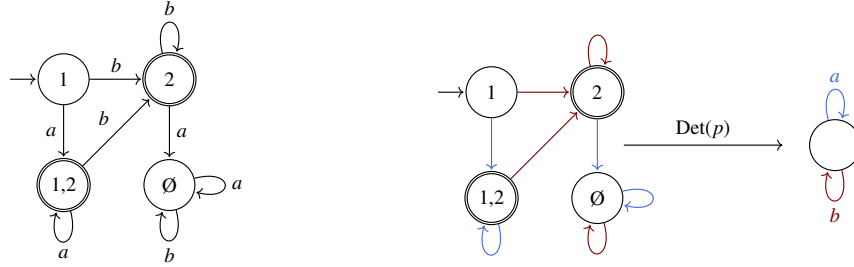
$$\begin{array}{ccccccc} \mathcal{C} & \xrightarrow{F} & \text{Span}(\text{Set}) & \xlongequal{\quad} & \text{Span}(\text{Set}) & \xlongequal{\quad} & \text{Span}(\text{Set}) \\ & & \searrow \text{Im} & \Uparrow \eta & \nearrow i & & \nearrow i \\ & & \text{Rel} & \xlongequal{\quad} & \text{Rel} & & \\ & & \searrow R & \Uparrow \varepsilon & \nearrow i & & \\ & & \text{Set} & & & & \end{array}$$

Moreover, it indeed recognizes the same language as postcomposing with Im does not affect the language and the analogous result for **Rel**-automata. Hence this determinization procedure is a fitting determinization procedure.

4.2 Universal Property for Span(Set)

Having proven the canonicity of our determinization, we now construct the universal property of the determinization of **Span(Set)**-automata. As before, we rely on bisimulations, which we need to adapt to **Span(Set)**. For this purpose, recall **Span(Set)** is a dagger category.

Using the determinization process presented in the previous section, the determinization of this automaton is the ULF-automaton pictured below to the right, which is equivalent to what we would get using the algorithmic procedure, pictured below to the left.

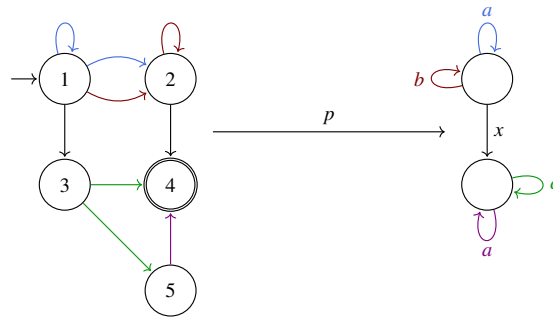


Let us recall the algorithmic procedure for determinizing an automaton. Given an automaton $M = (\Sigma, Q, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q), q_0, Q_f)$ we begin by taking the powerset of the set of states $\mathcal{P}(Q)$, this will be the states of our deterministic machine. Then for the transition function we just take the function $\delta' : \mathcal{P}(Q) \times \Sigma \rightarrow \mathcal{P}(Q)$, generated by δ , mapping $S \subseteq Q \mapsto \{q \in Q \mid \exists q'(q \in \delta(q'))\}$.

When we have a free monoid as base category for our automaton the determinization we obtain is analogous to the algorithmic one. This is because postcomposition by R replaces the objects in $\mathcal{C}$ with the powersets of their fibers, if we only have one fiber, then the determinization has its powerset as set of states.

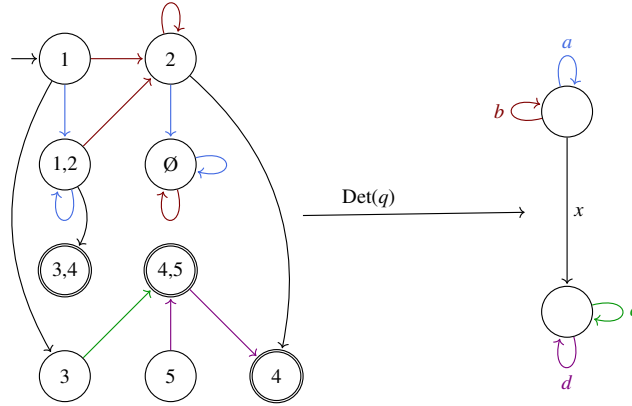
For a general base category we do not get the full powerset of the original set of states, as the states of our deterministic machine. Instead the determinization process takes to a certain extent into account which subsets would be unreachable. The limiting factor here is that these are only the subsets comprised of states lying over different fibers and consequently have no chance of a path to or from them.

Example 4.15. Consider the following automaton M , with ULF-functor p :



It recognizes the language $\{a^n b^m xcd\} \cup \{a^n b^m xc\}$, in particular has two "disjoint" alphabets $\{a, b\}$ and $\{c, d\}$ with a connective x , meaning that we have no paths combining them without an x in the middle, and once we have "put" the x after a possibly empty word $a^n b^m$ we cannot go back and only have $\{c, d\}$ available.

The following is the determinization of M using our determinization:



We see here that we do not get states containing states from different fibers, which makes our determinized automaton smaller than the powerset.

5 Multiset determinization for $\mathbf{Span}(\mathbf{Set})$ -automata

One step in the determinization process presented above that might feel unjustified is the identification of paths between states having the same label (i.e. postcomposition with the image functor into $\mathbf{Rel}$). To make amends for this we might consider a path-relevant version of determinization using multisets. It is worth noting that this significantly increases the size of our final machine, however, it strengthens the universal property.

5.1 Procedure

Multisets only correspond to finite spans, consequently we will restrict ourselves to $\mathbf{Span}(\mathbf{Fin})$ -automata. This is not unjustified as we in the definition of ULF-automaton have the condition that the fibers of the ULF-functor are finite, that is, its corresponding $\mathbf{Span}(\mathbf{Set})$ -functor factors through $\mathbf{Span}(\mathbf{Fin})$. To define the multiset determinization use that the multiset functor forms a relative monad $M : \mathbf{Fin} \rightarrow \mathbf{Set}$ and the fact that we for the "relative" Kleisli-category on $\mathbf{Fin}$, have that $\mathbf{Fin}_M(A, B)$ is the skeleton of $\mathbf{Span}(\mathbf{Fin})(A, B)$ for finite sets A and B . For a thorough treatment of the theory of relative monads we refer the reader to [2, 4].

Proposition 5.1. *Multisets form a relative monad from $\mathbf{Fin}$ to $\mathbf{Set}$, defined for all $A, B : \mathbf{Fin}$ and $\varphi : A \rightarrow \mathbb{N}^B$ by*

$$\begin{array}{lll} M : \mathbf{Fin} & \rightarrow & \mathbf{Set} \\ A & \mapsto & \mathbb{N}^A \end{array} \quad \begin{array}{lll} \eta_A : A & \rightarrow & \mathbb{N}^A \\ a & \mapsto & \sum_{a' \in A} \chi_{aa'} \end{array} \quad \begin{array}{lll} \varphi^* : \mathbb{N}^A & \rightarrow & \mathbb{N}^B \\ f & \mapsto & \sum_{a \in A} f(a) \cdot \varphi(a)(-) \end{array}$$

Lemma 5.2. *There is a local equivalence of categories between $\mathbf{Span}(\mathbf{Fin})$ and $\mathbf{Fin}_M$ that is identity on objects defined by*

$$\begin{array}{lll} U : \mathbf{Span}(\mathbf{Fin})(A, B) & \rightleftarrows & \mathbf{Fin}_M(A, B) : \Pi \\ A \xleftarrow{f} S \xrightarrow{g} B & \mapsto & (a \mapsto (b \mapsto |(f \times g)^{-1}(a, b)|)) \\ \langle \pi_1, \pi_2 \rangle : \Pi_{(a,b) \in A \times B} f(a)(b) \Rightarrow \langle A, B \rangle & \leftarrow & f : A \rightarrow \mathbb{N}^B \end{array}$$

Construction 5.3. Given a **Span(Fin)**-automaton $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Fin})$, its multiset determinization is given by the following, where $\mathbf{Fin}_M$ is the relative Kleisli-category associated to M .

$$\mathbf{MDet}(F) := \mathcal{C} \xrightarrow{F} \mathbf{Span}(\mathbf{Fin}) \xrightarrow{U} \mathbf{Fin}_M \xrightarrow{M} \mathbf{Set}$$

As $\mathbf{MDet}(M)$ is based on the multiset monad it can be seen as a determinization that remembers which states it passed and how many different ways it passed it.

We must again make sure that there is a canonical simulation between an automaton and its determinization, as well as the fact that it recognizes the same language. Here the use of the local equivalence of categories and relative Kleisli-category assures us of having a canonical simulation that is forward-backward, instead of just forward.

Lemma 5.4. *Let $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Fin})$ be an automaton, then there is a canonical forward-backward simulation from F to ΠUF with pseudo naturality given by the η of the local equivalence of categories.*

$$\begin{array}{ccc} \mathcal{C} & \xrightarrow{F} & \mathbf{Span}(\mathbf{Fin}) \\ & & \downarrow U \\ & & \mathbf{Fin}_M \\ & \uparrow \eta & \uparrow \Pi \\ & & \mathbf{Span}(\mathbf{Fin}) \end{array} \quad \begin{array}{ccc} A & \xrightarrow{\Pi_{(a,b) \in A \times B} S(a,b)} & B \\ \text{Id}_A \downarrow & \nearrow \eta & \downarrow \text{Id}_B \\ A & \xrightarrow{S} & B \end{array}$$

Lemma 5.5. *There is a natural transformation β , with the following naturality square (commuting in $\mathbf{Set}$)*

$$\begin{array}{ccc} \mathbf{Fin}_M & \xrightarrow{i} & \mathbf{Set}_M \\ & \downarrow U & \uparrow \beta \\ & & \mathbf{Set} \end{array} \quad \begin{array}{ccc} \mathbb{N}^A & \xrightarrow{\eta_{\mathbb{N}^B} f^*} & \mathbb{N}^{\mathbb{N}^B} \\ \text{Id}_{\mathbb{N}^A} \downarrow & & \downarrow \text{Id}_{\mathbb{N}^B}^* \\ \mathbb{N}^A & \xrightarrow{f^*} & \mathbb{N}^B \end{array}$$

Proposition 5.6. *Let $F : \mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Fin})$ be an automaton, then there is a canonical forward-backward simulation from F to $\mathbf{MDet}(F)$.*

$$\begin{array}{ccccc} \mathcal{C} & \xrightarrow{F} & \mathbf{Span}(\mathbf{Fin}) & \xrightarrow{i} & \mathbf{Span}(\mathbf{Set}) \\ & & \downarrow U & \uparrow \eta & \uparrow i \\ & & \mathbf{Fin}_M & \xrightarrow{i} & \mathbf{Set}_M \\ & & \downarrow M & \uparrow \beta & \uparrow i \\ & & \mathbf{Set} & & \end{array}$$

5.2 Universal Property

We again rely on constructing pasting inverses to obtain a universal property. To construct these we rely on the fact that our construction is based on a local equivalence of categories as well as a relative adjunction induced by a relative monad.

Lemma 5.7. *The natural transformation α is the identity when restricted to $\mathbf{Fin}_M$, as the ε from the local equivalence of categories Lemma 5.2 is the identity.*

Lemma 5.8. *The natural transformation β is the pasting inverse of the simulation η from the relative adjunction*

Using these lemmas we find the following corresponding universal property:

Proposition 5.9. *Let $G : \mathcal{C} \rightarrow \mathbf{Set}$ be some deterministic automaton, then any simulation from F to G factors through the canonical forward-backward simulation to $\mathbf{MDet}(F)$ and a bisimulation between $\mathbf{MDet}(F)$ and G .*

Proof. By Lemma 5.7 and Lemma 5.8 we have that the following diagram:

$$\begin{array}{ccc}
 \mathcal{C} & \xrightarrow{iF_p} & \mathbf{Span}(\mathbf{Set}) \\
 \downarrow G & \uparrow \alpha & \downarrow i\Pi \\
 & \mathbf{Fin}_M & \\
 & \uparrow i & \\
 & \mathbf{Fin} &
 \end{array}
 =
 \begin{array}{ccccc}
 \mathcal{C} & \xrightarrow{F_p} & \mathbf{Span}(\mathbf{Fin}) & \xrightarrow{\Pi} & \mathbf{Span}(\mathbf{Fin}) & \xrightarrow{i} & \mathbf{Span}(\mathbf{Set}) \\
 \downarrow G & \uparrow \alpha & \downarrow \Pi & \downarrow U & \downarrow \Pi & \downarrow i & \\
 & \mathbf{Fin}_M & \xrightarrow{M} & \mathbf{Fin}_M & \xrightarrow{i} & \mathbf{Set}_M & \\
 & \uparrow i & \uparrow \eta_i & \uparrow M & \uparrow \beta & \uparrow i & \\
 & \mathbf{Fin} & \xrightarrow{i} & \mathbf{Set} & \xrightarrow{i} & \mathbf{Set} &
 \end{array}$$

Consequently this gives a unique factorization of α through the universal simulation between F and $\mathbf{MDet}(F)$. $\square$

6 Conclusion

In this paper, we studied determinizations for **Rel**-automata and ULF-automata from a categorical point of view. Determinization for **Rel**-automata has previously been investigated by Colcombet and Petrişan in [5], as ULF-automata is a generalization of **Rel**-automata, through the $\mathbf{Span}(\mathbf{Set})$ intermediary, we consider how it might be generalized to $\mathbf{Span}(\mathbf{Set})$ -automata and thus also to ULF-automata.

To do so, we gave a detailed presentation of the determinization procedure for **Rel**, which relies on the Kleisli adjunction between **Set** and **Rel**, as well as its correctness and universal property as previously described by Colcombet and Petrişan.

Generalizing this construction to $\mathbf{Span}(\mathbf{Set})$ turned out to be non-trivial, as we neither have an adjunction between $\mathbf{Span}(\mathbf{Set})$ and **Set**, nor one between $\mathbf{Span}(\mathbf{Set})$ and **Rel**. Moreover, as $\mathbf{Span}(\mathbf{Set})$ -automata corresponds to pseudofunctors, it forced us to work in a 2-categorical setting and adapt definitions to it. We have shown that it is possible to recover a fitting determinization procedure by first turning a pseudo-functor $\mathcal{C} \rightarrow \mathbf{Span}(\mathbf{Set})$ into $\mathcal{C} \rightarrow \mathbf{Rel}$ using the local adjunction and between $\mathbf{Span}(\mathbf{Set})$ and **Rel** then using the **Rel**-determinization to get back a functor $\mathcal{C} \rightarrow \mathbf{Set}$. To prove correctness and equip the procedure with a similar universal property we used the local adjunction and the correctness and universal property for **Rel**-determinization. The determinized automaton we obtained has subsets of states as states and a transition function, which given a letter maps a subset to the set of reachable states in the original automaton, analogously to usual algorithmic determinization procedure. We showed that our procedure does not always get whole powerset as the set of states, but does as soon as we consider automata with a free monoid as codomain.

This determinization procedure has the particularity to identify the paths taken, as this may not be wanted, we provided an alternative determinization based on the multiset relative monad that retains them. Compared to the one before, we no longer rely on the **Rel** determinization procedure as a middle step, as factoring through **Rel** identifies paths. Instead, we use the relative Kleisli category of the relative multiset monad M , $\mathbf{Fin}_M$, as a middle step, and then postcompose by the functor from $\mathbf{Fin}_M$ to **Set** provided by Kleisli construction of $\mathbf{Fin}_M$. Moreover, we have proven the correctness of this procedure and provided it with a universal property, leveraging the relative monad and that there is a local equivalence of categories between $\mathbf{Fin}_M$ and $\mathbf{Span}(\mathbf{Fin})$. The multiset determinization enabled us to achieve path relevance, and to obtain a stronger universal property that encompasses forward-backward simulation, as we now get a canonical forward-backward simulation between an automaton F and its multiset determinization $\mathbf{MDet}(F)$.

There are many potential ways to expand the work on categorical automata, one potential possibility is to explore the multiset determinization from the point of view of transducers where path relevance might be desired as we in addition to the input also have an output to take into account. Another possibility would be minimization, as this has been done for **Rel**-automata, we might consider how to generalize this to our setting.

Acknowledgements

The work for this paper was done during an summer internship under the supervision of Noam Zeilberger, the author would thus like to thank him for the introduction to the subject and many fruitful discussions. The author is also grateful to Thomas Lamiaux for his advice on writing and his comments on the many drafts of the paper. Finally, the author would like to thank the anonymous reviewers for their comments and suggestions.

References

- [1] J. Adámek & V. Trnková (1990): *Automata and Algebras in Categories*. Mathematics and its Applications, Springer Netherlands. Available at https://books.google.fr/books?id=MkADy_WxInsC.
- [2] Thosten Altenkirch, James Chapman & Tarmo Uustalu (2015): *Monads need not be endofunctors*. *Logical Methods in Computer Science* Volume 11, Issue 1, doi:10.2168/lmcs-11(1:3)2015.
- [3] Michael A. Arbib & Ernest G. Manes (1980): *Machines in a category*. *Journal of Pure and Applied Algebra* 19, pp. 9–20, doi:10.1016/0022-4049(80)90090-0.
- [4] Nathanael Arkor & Dylan McDermott (2024): *The formal theory of relative monads*. *Journal of Pure and Applied Algebra* 228(9), p. 107676, doi:10.1016/j.jpaa.2024.107676.
- [5] Thomas Colcombet & Daniela Petrişan (2020): *Automata Minimization: a Functorial approach*. *Logical Methods in Computer Science*, pp. 1–32, doi:10.23638/LMCS-16(1:32)2020. Available at <https://hal.science/hal-03105616>.
- [6] Samuel Eilenberg & Jesse B. Wright (1967): *Automata in general algebras*. *Information and Control* 11(4), pp. 452–470, doi:10.1016/S0019-9958(67)90670-5.
- [7] Paul-André Melliès & Noam Zeilberger (2022): *Parsing as a lifting problem and the Chomsky-Schützenberger representation theorem*. In: *MFPS 2022 - 38th conference on Mathematical Foundations for Programming Semantics*, Ithaca, NY, United States. Available at <https://hal.science/hal-03702762>.
- [8] Paul-André Melliès & Noam Zeilberger (2023): *The categorical contours of the Chomsky-Schützenberger representation theorem*. Available at <https://hal.science/hal-04399404>. This is a thoroughly revised and expanded version of a paper with a similar title (hal-03702762, arXiv:2212.09060) presented at the 38th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2022). 62 pages, including a 13 page Addendum on “gCFLs as initial models of gCFGs”, and a table of contents.
- [9] M. O. Rabin & D. Scott (1959): *Finite Automata and Their Decision Problems*. *IBM Journal of Research and Development* 3(2), pp. 114–125, doi:10.1147/rd.32.0114.
- [10] J.J.M.M. Rutten (2000): *Universal coalgebra: a theory of systems*. *Theoretical Computer Science* 249(1), pp. 3–80, doi:10.1016/S0304-3975(00)00056-6. Modern Algebra.
- [11] Paweł Sobociński (2012): *Relational Presheaves as Labelled Transition Systems*. In Dirk Pattinson & Lutz Schröder, editors: *11th International Workshop on Coalgebraic Methods in Computer Science (CMCS), Coalgebraic Methods in Computer Science LNCS-7399*, Springer, Tallinn, Estonia, pp. 40–50, doi:10.1007/978-3-642-32784-1_3. Available at <https://inria.hal.science/hal-01539889>.

Generalized Gradient Descent is a Hypergraph Functor

Tyler Hanks

Department of Computer and Information Science and Engineering
University of Florida
Gainesville, Florida

t.hanks@ufl.edu

James Fairbanks

fairbanksj@ufl.edu

Matthew Klawonn

Information Directorate
Air Force Research Lab
Rome, New York

matthew.klawonn.2@us.af.mil

Cartesian reverse derivative categories (CRDCs) provide an axiomatic generalization of the reverse derivative, which allows generalized analogues of classic optimization algorithms such as gradient descent to be applied to a broad class of problems. In this paper, we show that generalized gradient descent with respect to a given CRDC induces a hypergraph functor from a hypergraph category of optimization problems to a hypergraph category of dynamical systems. The domain of this functor consists of objective functions that are 1) general in the sense that they are defined with respect to an arbitrary CRDC, and 2) open in that they are decorated spans that can be composed with other such objective functions via variable sharing. The codomain is specified analogously as a category of general and open dynamical systems for the underlying CRDC. We describe how the hypergraph functor induces a distributed optimization algorithm for arbitrary composite problems specified in the domain. To illustrate the kinds of problems our framework can model, we show that parameter sharing models in multitask learning, a prevalent machine learning paradigm, yield a composite optimization problem for a given choice of CRDC. We then apply the gradient descent functor to this composite problem and describe the resulting distributed gradient descent algorithm for training parameter sharing models.

1 Introduction

Cartesian reverse derivative categories (CRDCs) provide an axiomatic generalization of the reverse derivative and have been used to define generalized analogues of classic optimization algorithms, such as gradient descent, for minimizing generalized objective functions [7, 16]. These developments allow techniques from machine learning such as backpropagation to be applied not only to artificial neural networks, but to a broad class of models including boolean circuits. The generality of the CRDC framework makes future expansion to other problem types likely. Central to the definition of a CRDC is the Cartesian reverse derivative combinator $\mathbf{R}$, which must obey axioms mirroring the behavior of the standard directional derivative in the Euclidean domain. This combinator is *compositional* in that it satisfies a chain rule, allowing one to define a general version of the backpropagation algorithm [14].

In prior work, CRDCs have been used primarily to compose parameterized morphisms, i.e to build a learning model, all while leveraging $\mathbf{R}$ to ensure the parameters can be updated with respect to the generalized gradient of an objective. In this way, the induced optimization problem of maximizing/minimizing the given objective with respect to the parameters has a compositional structure by virtue of the *model* being compositional. We propose that in addition to compositional structure in the *model*, compositional structure in the *objective* is also of interest. In particular, it is common in machine learning for parameters to be optimized for more than one objective. Simple examples of compositional objectives are those arising from various regularization methods, for example ℓ_1 or ℓ_2 penalties [3], wherein one term in the objective incentivizes “good performance” on the given task while the other enforces some generically desirable property like sparseness. Other examples with more sophisticated composition patterns

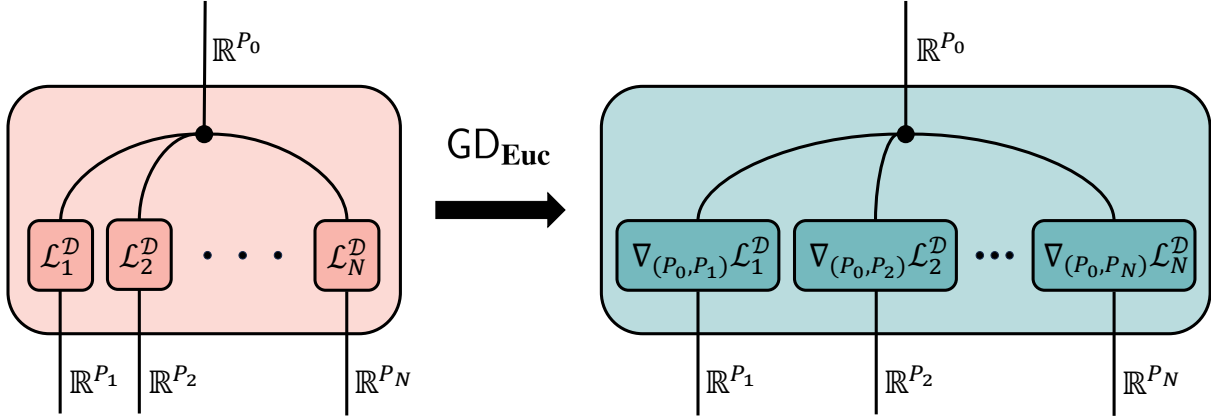


Figure 1: An illustration of multitask learning in our framework. The left string diagram shows the composite objective function representing the multitask learning objective. The right string diagram is the result of applying the gradient descent functor to the morphism on the left, resulting in a distributed optimization algorithm.

include example- and feature-parallel distributed learning setups [4], and parameter sharing methods wherein subsets of parameters may be optimized for one or more objective functions. Parameter sharing arises within the setting of multitask learning, which we will discuss in this paper. Figure 1 illustrates this example. Our goal in this paper is to formalize compositional objective structure and its exploitation via gradient descent in the general setting of CRDCs.

To capture composition of objective functions that may have shared parameters, we turn to decorated spans and hypergraph categories, which are useful for formalizing mathematical objects that compose in an undirected, networked fashion [9]. Our approach closely follows that of [11], where algebras of undirected wiring diagrams are used to compose objective functions with shared decision variables, and a variety of first-order methods including (sub)gradient descent and primal-dual methods are shown to be functorial in the standard Euclidean domain. This paper extends the results on gradient descent to the general setting of Cartesian reverse derivative categories and generalized optimization introduced in [16]. Specifically, our contributions are as follows.

- We show how to produce hypergraph categories $\mathbf{Opt}_C^R$ of generalized open objectives and $\mathbf{Dynam}_C$ of optimizers over a given CRDC C and optimization domain (C, R) .
- We prove that generalized gradient descent yields a hypergraph functor

$$\mathbf{GD}_C : \mathbf{Opt}_C^R \rightarrow \mathbf{Dynam}_C$$

from open objectives to open optimizers.

- We highlight how functoriality allows the gradient descent solution algorithms for composite problems to be implemented in a distributed fashion.
- We show that multitask learning [5] with hard parameter sharing is an example of objective composition occurring in the machine learning literature, and we leverage our functorial formulation of gradient descent to recover a distributed training scheme.

After providing the necessary background on CRDCs, we will begin by setting up the categories of open generalized objectives and open dynamical systems. We then specify the generalized gradient descent functor between them before turning to the multitask learning example.

2 Preliminaries

In this section, we recall the necessary definitions and concepts from Cartesian reverse derivative categories [6] and generalized optimization [16].

Definition 2.1 (Definition 3.1 in [16]). A **Cartesian left-additive category** is a Cartesian category $\mathcal{C}$ with terminal object $*$ in which for any pair of objects X, Y , the hom-set $\mathcal{C}(X, Y)$ is a commutative monoid with addition operation $+$ and additive identities $0_{X,Y}: X \rightarrow Y$. In addition, the following axioms must be satisfied.

- **LA.1** For any morphisms $h: X \rightarrow Y$ and $f, g: Y \rightarrow Z$, we have

$$(f + g) \circ h = (f \circ h) + (g \circ h): X \rightarrow Z \text{ and } 0_{Y,Z} \circ h = 0_{X,Z}: X \rightarrow Z. \quad (1)$$

- **LA.2** For any projection map $\pi_i: Y \rightarrow Z$ and morphisms $f, g: X \rightarrow Y$ we have

$$\pi_i \circ (f + g) = (\pi_i \circ f) + (\pi_i \circ g): X \rightarrow Z \text{ and } \pi_i \circ 0_{X,Y} = 0_{X,Z}: X \rightarrow Z. \quad (2)$$

The additive identity of $\mathcal{C}(*, X)$ is denoted 0_X .

Rough Definition 2.2 (Definition 3.3 in [16]). A **Cartesian differential category** is a Cartesian left-additive category $\mathcal{C}$ equipped with a **Cartesian derivative combinator** $\mathbf{D}$ taking morphisms $f: X \rightarrow Y$ to morphisms $\mathbf{D}[f]: X \times X \rightarrow Y$. This combinator must satisfy several axioms requiring that it behave like a standard forward derivative.

Rough Definition 2.3 (Definition 3.2 in [16]). A **Cartesian reverse derivative category** is a Cartesian left-additive category $\mathcal{C}$ equipped with a **Cartesian reverse derivative combinator** $\mathbf{R}$ taking morphisms $f: X \rightarrow Y$ to morphisms $\mathbf{R}[f]: X \times Y \rightarrow X$. This combinator must satisfy several axioms requiring that it behave like a standard reverse derivative. In particular, we will make use of the following axioms in this paper.

- **RD.1** $\mathbf{R}[f + g] = \mathbf{R}[f] + \mathbf{R}[g]$ and $\mathbf{R}[0] = 0$.
- **RD.5** $\mathbf{R}[g \circ f] = \mathbf{R}[f] \circ (\text{id} \times \mathbf{R}[g]) \circ \langle \pi_0, \langle f \circ \pi_0, \pi_1 \rangle \rangle$.

Intuitively, **RD.1** says that the reverse derivative combinator must be additive while **RD.5** is the chain rule cast in the abstract setting of CRDCs. Every Cartesian reverse derivative category (CRDC) is also a Cartesian differential category (Theorem 16 in [6]) by defining the derivative combinator of a morphism $f: X \rightarrow Y$ as

$$\mathbf{D}[f] = \pi_1 \circ \mathbf{R}[\mathbf{R}[f]] \circ (\langle \text{id}_X, 0_{X,Y} \rangle \times \text{id}_X): X \times X \rightarrow Y. \quad (3)$$

This allows us to refer to the Cartesian derivative combinator of a CRDC. Of crucial importance in the subsequent sections is the notion of a linear map in a CRDC.

Definition 2.4. A **linear map** in a CRDC $\mathcal{C}$ is a morphism f such that $\mathbf{D}[f] = f \circ \pi_1$. The linear maps in $\mathcal{C}$ form a full subcategory of $\mathcal{C}$ which we denote $\text{Lin}(\mathcal{C})$. By Proposition 24 in [6], $\text{Lin}(\mathcal{C})$ is a $\dagger$ -category with finite $\dagger$ -biproducts. Moreover, $\mathbf{R}[f] = f^\dagger \circ \pi_1$ and $(f + g)^\dagger = f^\dagger + g^\dagger$.

There are two canonical examples of CRDCs in the literature.

1. The CRDC **Euc** has Euclidean spaces as objects and smooth functions as morphisms. The terminal object is $\mathbb{R}^0$ and the reverse derivative of a smooth function $f: \mathbb{R}^m \rightarrow \mathbb{R}^n$ is

$$\mathbf{R}[f](x, x') := J_f(x)^T x',$$

where $J_f(x)$ is the Jacobian of f evaluated at x . The linear maps in **Euc** are the usual linear maps between vector spaces, with the dagger structure corresponding to dual maps.

2. Given a commutative ring r , the CRDC **Poly_r** has natural numbers as objects. A morphism from n to m is an m -tuple of polynomials over r , each in n variables. The terminal object is 0 and the reverse derivative of a polynomial $P(x) = (p_1(x), \dots, p_m(x))$ is

$$\mathbf{R}[P](x, x') := \left(\sum_{i=1}^m \frac{\partial p_i}{\partial x_1}(x) x'_i, \dots, \sum_{i=1}^m \frac{\partial p_i}{\partial x_n}(x) x'_i \right),$$

where $\frac{\partial p_i}{\partial x_j}(x)$ denotes the formal derivative of p_i in x_j at x . The linear maps are polynomials $P(x) = (p_1(x), \dots, p_m(x))$ where each $p_i(x)$ is of the form $p_i(x) = \sum_{i=1}^n r_i x_i$ for $r_i \in r$ [6].

The fundamental component for generalized optimization is the notion of an *optimization domain*, which generalizes the role of the real numbers in defining the “cost” or “value” of a given configuration of decision variables.

Definition 2.5 (Definition 3.4 in [16]). An **optimization domain** is a pair $(\mathcal{C}, R)$ where $\mathcal{C}$ is a CRDC and R is an object of $\mathcal{C}$ equipped with the following additional structures.

- Each morphism $f: X \rightarrow Y$ in $\mathcal{C}$ has an additive inverse $-f$.
- Each hom-set $\mathcal{C}(*, X)$ out of the terminal object is equipped with a multiplication operation fg and a multiplicative identity $1_X: * \rightarrow X$ to form a commutative ring with the left additive structure.
- The hom-set $\mathcal{C}(*, R)$ is totally ordered to form an ordered commutative ring.

Given a unique map $!_X: X \rightarrow *$ into the terminal object, the map $1_Y \circ !_X: X \rightarrow Y$ is denoted $1_{X,Y}$. An **objective** in $(\mathcal{C}, R)$ is a morphism $\ell: X \rightarrow R$ in $\mathcal{C}$.

Definition 2.6 (Definition 3.6 in [16]). Given an optimization domain $(\mathcal{C}, R)$, the **generalized gradient** of an objective $\ell: X \rightarrow R$ is $\mathbf{R}[\ell]_1: X \rightarrow X$ defined by

$$\mathbf{R}[\ell]_1 := \mathbf{R}[\ell] \circ \langle \text{id}_X, 1_{X,R} \rangle. \quad (4)$$

The following are examples given in [16] of optimization domains and generalized gradients.

1. The **standard domain** is $(\mathbf{Euc}, \mathbb{R})$. Objectives are smooth functions $\ell: \mathbb{R}^n \rightarrow \mathbb{R}$ and the gradient of $\ell: \mathbb{R}^n \rightarrow \mathbb{R}$ is $\nabla \ell: \mathbb{R}^n \rightarrow \mathbb{R}^n$.
2. The **r -polynomial domain** is $(\mathbf{Poly}_r, 1)$. Objectives are polynomials $\ell: n \rightarrow 1$ and the gradient of $\ell: n \rightarrow 1$ is $\left(\frac{\partial \ell}{\partial x_1}(x), \dots, \frac{\partial \ell}{\partial x_n}(x) \right)$, where these are again formal derivatives.

3 Generalized Gradient Descent as a Hypergraph Functor

To construct hypergraph categories of objectives and optimizers, we will require that the subcategory of linear maps in $\mathcal{C}$, denoted $\text{Lin}(\mathcal{C})$, has finite limits. Going forward, we use $\times$ to denote the product and $*$ to denote the terminal object in an arbitrary CRDC. Proofs of theorems are given in the appendix.

3.1 The Hypergraph Category of Open Objectives

The goal of this section is to define a hypergraph category of generalized optimization problems. Recall that a hypergraph category is a symmetric monoidal category where every object is equipped with the structure of a *special commutative Frobenius monoid* [10]. In particular, this means that every object has the structure of both a commutative monoid and a cocommutative comonoid and satisfies two additional axioms to ensure that these structures cohere properly. It can be further shown from these axioms that every hypergraph category is self-dual compact closed. Taken together, these properties yield a category where morphisms can compose in an undirected fashion according to the interconnection pattern of arbitrary hypergraphs (hence the name hypergraph categories). Another way of saying this is that hypergraph categories are symmetric monoidal categories whose string diagrams (introduced in [12]) are hypergraphs. We now turn to how this structure is realized in optimization problems.

In standard optimization, a generic unconstrained minimization problem has the form

$$\text{minimize } f(x),$$

where $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is known as the objective function, and $x \in \mathbb{R}^n$ is referred to as the decision or optimization variable. When $f(x)$ has the form $\sum_{i=1}^N f_i(x_i)$, where the x_i are disjoint subvectors of x , f is referred to as *separable*. Separable objectives can be solved entirely in parallel as each optimal x_i has no effect on the rest of the objective. However, in many important problems, the objective function is *partially separable*. For example, the problem

$$\text{minimize } f(w, x) + g(u, w, y) + h(u, w, z)$$

consists of a sum of objectives with *complicating* variables u and w . Such a sum of objectives with shared variables is the type of compositional structure that we wish to capture and generalize in our hypergraph category of optimization problems.

A standard method for defining such a hypergraph category is to leverage the theory of decorated spans¹ [8, 9]. Decorated spans turn a closed system into an open system by specifying a boundary for the system in addition to morphisms that relate the domain of the system to its boundary. We now leverage this general pattern to define a notion of *open objectives*.

Definition 3.1. Given an optimization domain $(\mathcal{C}, R)$, an **open objective** with domain boundary X and codomain boundary Y consists of a span $X \leftarrow N \rightarrow Y$ in $\text{Lin}(\mathcal{C})$ together with a choice of objective on N , i.e., a map $f: N \rightarrow R$ in $\mathcal{C}$. This is summarized by the diagram

$$\begin{array}{ccc} & R & \\ & \uparrow f & \\ X & \xleftarrow{l} N \xrightarrow{r} & Y. \end{array}$$

There is a straightforward interpretation of open objectives when the legs are both projections: the boundary objects are subobjects of the overall decision space which can be shared with other objectives. This intuition of sharing parts of the decision space is still helpful even when the legs are general linear maps. The theory of decorated spans provides a recipe for constructing hypergraph categories of such open systems by specifying a lax symmetric monoidal functor from a finitely complete category into **Set**, known as a *decorating functor*. We thus arrange open objectives into a hypergraph category by specifying the following decorating functor.

¹The literature typically uses *covariant* decorating functors to define decorated cospan categories. However, all the same results hold for the dual construction of *contravariant* decorating functors and decorated spans, which we leverage in this paper.

Theorem 3.2. *Given an optimization domain $(\mathcal{C}, R)$, there is a contravariant lax symmetric monoidal functor $O_{\mathcal{C}}^R: (\text{Lin}(\mathcal{C})^{\text{op}}, \times) \rightarrow (\mathbf{Set}, \times)$ defined by the following maps (where we let $O := O_{\mathcal{C}}^R$ for the remainder of the theorem and proof).*

- *Given an object X , $O(X)$ is the set of generalized objectives with domain X , i.e., the hom-set $\mathcal{C}(X, R)$.*
- *Given a linear map $\phi: X \rightarrow Y$ and objective $f: Y \rightarrow R$, $O(\phi)(f): X \rightarrow R$ is the objective $f \circ \phi$ obtained by precomposition.*
- *Given objects X, Y , the product comparison $\varphi_{X,Y}: O(X) \times O(Y) \rightarrow O(X \times Y)$ is defined by*

$$\varphi_{X,Y}(f, g) := f \circ \pi_0 + g \circ \pi_1, \quad (5)$$

where $\pi_0: X \times Y \rightarrow X$ and $\pi_1: X \times Y \rightarrow Y$ are the natural projections.

- *The unit comparison $\varphi_0: \{*\} \rightarrow O(*)$ selects the additive identity 0_R for the hom-set $\mathcal{C}(*, R)$ supplied by the left additive structure.*

We can now use our decorated spans of open objectives to construct a hypergraph category.

Corollary 3.2.1 (Open Generalized Optimization Problems). *Given an optimization domain $(\mathcal{C}, R)$ such that $\text{Lin}(\mathcal{C})$ has finite limits, there is a hypergraph category $\mathbf{Opt}_{\mathcal{C}}^R$ defined by the following data.*

- *Objects are the same as those of $\mathcal{C}$.*
- *Morphisms are (isomorphism classes of) open generalized objectives.*
- *Given two open objectives $(X \xleftarrow{l_1} N \xrightarrow{r_1} Y, f \in O_{\mathcal{C}}^R(N))$ and $(Y \xleftarrow{l_2} M \xrightarrow{r_2} Z, g \in O_{\mathcal{C}}^R(M))$, their composite consists of the following span computed by pullback*

$$\begin{array}{ccccc}
 & & N \times_Y M & & \\
 & \swarrow b_0 & \downarrow \langle b_0, b_1 \rangle & \searrow b_1 & \\
 N & \xleftarrow{\pi_0} & N \times M & \xrightarrow{\pi_1} & M \\
 \swarrow l_1 & & \downarrow r_1 & & \downarrow r_2 \\
 X & & Y & & Z
 \end{array} \quad (6)$$

together with the objective obtained from the composite morphism

$$\{*\} \cong \{*\} \times \{*\} \xrightarrow{f \times g} O_{\mathcal{C}}^R(N) \times O_{\mathcal{C}}^R(M) \xrightarrow{\varphi_{N,M}} O_{\mathcal{C}}^R(N \times M) \xrightarrow{O_{\mathcal{C}}^R \langle b_0, b_1 \rangle} O_{\mathcal{C}}^R(N \times_Y M). \quad (7)$$

- *The identity on X is the pair $(X = X = X, \{*\} \xrightarrow{\varphi_0} O_{\mathcal{C}}^R(*) \xrightarrow{O_{\mathcal{C}}^R(!)} O_{\mathcal{C}}^R(X))$.*
- *The monoidal product on objects is their product in $\mathcal{C}$ while the monoidal product of morphisms $(X \leftarrow N \rightarrow Y, f)$ and $(W \leftarrow M \rightarrow Z, g)$ is*

$$(X \times W \leftarrow N \times M \rightarrow Y \times Z, \varphi_{N,M}(f, g)). \quad (8)$$

- *The hypergraph maps are inherited from those of $\text{Span}(\text{Lin}(\mathcal{C}))$ together with the identity decoration.*

Proof. This follows by applying Proposition 3.2 and Theorem 3.4 in [8] to Theorem 3.2. $\square$

3.2 The Hypergraph Category of Open Optimizers

Having defined a hypergraph category of open objective functions, we will now use the same machinery of decorated spans to define a hypergraph category of open dynamical systems over a given CRDC.

Definition 3.3. Given a CRDC $\mathcal{C}$, an **open dynamical system** with domain boundary X and codomain boundary Y consists of a span $X \leftarrow N \rightarrow Y$ in $\text{Lin}(\mathcal{C})$ together with a choice of system on N , i.e., a map $v: N \rightarrow N$ in $\mathcal{C}$.

Endomaps in a CRDC are also referred to as optimizers in [16], thus we use the terms optimizer and dynamical system interchangeably. Similar to open objectives, the boundaries of open dynamical systems specify which parts of a system's state space can be influenced by other systems. The decorating functor is given as follows.

Theorem 3.4. *Given a CRDC $\mathcal{C}$, there is a contravariant lax symmetric monoidal functor $D_{\mathcal{C}}: (\text{Lin}(\mathcal{C})^{\text{op}}, \times) \rightarrow (\mathbf{Set}, \times)$ defined by the following maps (where we let $D := D_{\mathcal{C}}$ for the remainder of the theorem and proof).*

- Given an object X , $D(X)$ is the set of generalized dynamical systems with state space X , i.e., the hom-set $\mathcal{C}(X, X)$.
- Given a linear map $\phi: X \rightarrow Y$ and a system $v: Y \rightarrow Y$, $D(\phi)(v)$ is the system $\phi^{\dagger} \circ v \circ \phi: X \rightarrow X$.
- Given objects X, Y , the product comparison $\phi_{X,Y}: D(X) \times D(Y) \rightarrow D(X \times Y)$ is defined by

$$\phi_{X,Y}(v, w) := \pi_0^{\dagger} \circ v \circ \pi_0 + \pi_1^{\dagger} \circ w \circ \pi_1. \quad (9)$$

- The unit comparison $\phi_0: \{*\} \rightarrow D(*)$ is uniquely determined as $\mathcal{C}(*, *)$ is singleton.

This is a generalization of the dynamics decorating functor defined in [1]. We can now define an analogous hypergraph category of open generalized dynamical systems.

Corollary 3.4.1 (Open Generalized Dynamics). *Given a CRDC $\mathcal{C}$ such that $\text{Lin}(\mathcal{C})$ has finite limits, there is a hypergraph category $\mathbf{Dynam}_{\mathcal{C}}$ defined by the following data.*

- Objects are the same as those of $\mathcal{C}$.
- Morphisms are (isomorphism classes of) open dynamical systems.
- Given two open systems $(X \xleftarrow{l_1} N \xrightarrow{r_1} Y, v \in D_{\mathcal{C}}(N))$ and $(Y \xleftarrow{l_2} M \xrightarrow{r_2} Z, w \in D_{\mathcal{C}}(M))$, their composite consists of the pullback span as in (6) together with the system obtained from the composite morphism

$$\{*\} \cong \{*\} \times \{*\} \xrightarrow{v \times w} D_{\mathcal{C}}(N) \times D_{\mathcal{C}}(M) \xrightarrow{\varphi_{N,M}} D_{\mathcal{C}}(N \times M) \xrightarrow{D_{\mathcal{C}}(b_0, b_1)} D_{\mathcal{C}}(N \times_Y M). \quad (10)$$

- The identity on X is the pair $(X = X = X, \{*\} \xrightarrow{\phi_0} D_{\mathcal{C}}(*) \xrightarrow{D_{\mathcal{C}}(!)} D_{\mathcal{C}}(X))$.
- The monoidal product on objects is their product in $\mathcal{C}$ while the monoidal product of morphisms $(X \leftarrow N \rightarrow Y, v)$ and $(W \leftarrow M \rightarrow Z, w)$ is

$$(X \times W \leftarrow N \times M \rightarrow Y \times Z, \varphi_{N,M}(v, w)). \quad (11)$$

- The hypergraph maps are inherited from those of $\text{Span}(\text{Lin}(\mathcal{C}))$ together with the identity decoration.

Proof. This follows by applying Proposition 3.2 and Theorem 3.4 in [8] to Theorem 3.4. $\square$

3.3 Functoriality of Generalized Gradient Descent

The pieces are now in place for our main result. We would like to show that one can functorially relate a composite objective function defined in $\mathbf{Opt}_C^R$ to a composite optimizer in $\mathbf{Dynam}_C$ that performs gradient descent on the given objective function. The framework of decorated spans allows the construction of such a hypergraph functor by specifying a monoidal natural transformation between the underlying decorating functors defining the domain and codomain hypergraph categories. Note that below we work with continuous gradient flow systems to simplify proofs. This is acceptable as gradient descent is equivalent to Euler's discretization method applied to a gradient flow system. Therefore, these results extend to discrete gradient descent systems because Euler's method is provably functorial in the Euclidean domain [13], and this proof can be easily extended to the general CRDC domain.

Theorem 3.5. *Given an optimization domain (C, R) , there is a monoidal natural transformation $\text{gd}(C): \mathbf{Opt}_C^R \Rightarrow \mathbf{Dynam}_C$ with components $\text{gd}(C)_X: \mathbf{Opt}_C^R(X) \rightarrow \mathbf{Dynam}_C(X)$ defined by the generalized gradient descent optimization scheme, i.e.,*

$$f \mapsto -\mathbf{R}[f]_1. \quad (12)$$

Corollary 3.5.1 (Functoriality of Generalized Gradient Flow). *Given an optimization domain (C, R) such that $\text{Lin}(C)$ has finite limits, there is an identity on objects hypergraph functor $\text{GD}_C: \mathbf{Opt}_C^R \rightarrow \mathbf{Dynam}_C$ which takes an open objective $(X \xleftarrow{l} N \xrightarrow{r} Y, f: N \rightarrow R)$ to the open optimizer $(X \xleftarrow{l} N \xrightarrow{r} Y, -\mathbf{R}[f]_1: N \rightarrow N)$.*

Proof. This follows by applying Theorem 4.1 of [8] to the monoidal natural transformation gd . $\square$

Taking stock, there are many implications of the categories and functor just defined. For one, corollary 3.5.1 says we have the following equality given any composable pair $F := (X \leftarrow N \rightarrow Y, f: N \rightarrow R), G := (Y \leftarrow M \rightarrow Z, g: M \rightarrow R)$ of open objectives:

$$\text{GD}_C(G \circ F) = \text{GD}_C(G) \circ \text{GD}_C(F). \quad (13)$$

In particular, this yields the following equivalence of open optimizers:

$$-\mathbf{R}[f \circ \pi_0 \circ \phi + g \circ \pi_1 \circ \phi]_1 = -\phi^\dagger \circ (\iota_0 \circ \mathbf{R}[f]_1 \circ \pi_0 + \iota_1 \circ \mathbf{R}[g]_1 \circ \pi_1) \circ \phi, \quad (14)$$

where we are using ϕ to denote the pairing $\langle b_0, b_1 \rangle$ from (6). Although these two optimizers are *extensionally* equivalent (i.e., they compute the same output for a given input), they are not *computationally* equivalent. In particular, the optimizer on the RHS of (14) is far better suited to implementation in a distributed computing environment as it can be interpreted with message passing semantics. Specifically, the distribute step is given by applying the linear map ϕ to the input followed by the application of the projections to send the desired components to the subsystems. The parallel computation step is given by computing the gradients of each objective with respect to their current local values, and the collect step is given by injecting the results into a single vector and applying the linear map $\phi^\dagger$.

Another benefit of open objectives and optimizers is the ability to easily specify composite objective functions using the graphical syntax of string diagrams. In the following section we will model hard parameter sharing for multi-task learning in our category $\mathbf{Opt}_C^R$. We do so to show that, rather than existing as an exercise in abstraction, our framework is applicable to a prevalent machine learning paradigm.

4 Multitask Learning

Multitask learning (MTL) [5] is a machine learning paradigm in which learning signals from different tasks are aggregated in order to train a single model. Precise definitions of what a task is vary, but for our purposes we will consider a task $\mathcal{T}_i := (\mathcal{D}_i, \mathcal{L}_i)$ to be a pair of data set $\mathcal{D}_i$ and loss function $\mathcal{L}_i$. Usually (see [19]) the tasks $\mathcal{T}_i$ are supervised, so $\mathcal{D}_i := \mathcal{X}_i \times \mathcal{Y}_i$ for input data $\mathcal{X}_i$ and labels $\mathcal{Y}_i$. The primary (or at least original) motivation of MTL is to exploit additional learning signals for improved model performance on a given “main” task, beyond what might be possible if the model were trained on any single task [5].

A common approach (see [19]) to multitask learning is to leverage parameterised models where at least some parameters are trained on all tasks. Such an approach is called *hard parameter sharing* because the shared parameters are updated *directly* using gradient information from multiple tasks. This is in contrast to *soft parameter sharing* where the parameters are only *penalized* for differing from counterparts trained on other tasks. Theoretically, hard parameter sharing helps to avoid overfitting of shared parameters [2]. When the underlying model is a neural network, hard parameter sharing is often achieved by sharing the parameters of the first several layers while training a unique set of predictive final layers per task. However, recent work suggests that more flexible schemes may lead to improved performance [18]. With this motivation, we will now frame hard parameter sharing as a composite optimization problem that allows arbitrary parameters to be shared amongst arbitrary tasks.

For each task $\mathcal{T}_i$ in a collection of N tasks, consider a neural network f_i that has task-specific parameters P_i and shared parameters P_0 . We will call an assignment of parameters to real numbers $W_i \in \mathbb{R}^{P_i}$ the *weights* of a neural network, though note that the machine learning literature often uses the terms “weights” and “parameters” interchangeably. The composite optimization problem given by such a multitask learning setup (cf. [15], (1)) is

$$\min_{W_0, \dots, W_N} \sum_{i=1}^N \mathcal{L}_i^D(f_i; W_0, W_i) \quad (15)$$

where

$$\mathcal{L}_i^D(f_i; W_0, W_i) := \frac{1}{|\mathcal{D}_i|} \sum_{d=1}^{|\mathcal{D}_i|} \mathcal{L}_i(f_i(\mathcal{X}_i^d; W_0, W_i), \mathcal{Y}_i^d) \quad (16)$$

with $f_i(\mathcal{X}_i^d; W_0, W_i)$ denoting the prediction of the network f_i weighted by (W_0, W_i) on input datum $\mathcal{X}_i^d$. We will now describe how the problem in (15) is a composite of open objectives, and therefore how to mechanically recover a distributed gradient descent algorithm for it.

4.1 Compositional Formulation

We will take our CRDC $\mathcal{C}$ to be **Euc** with objects euclidean spaces $\mathbb{R}^N$ and morphisms $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$ the smooth maps between spaces. The monoidal product is given by $\times$, the standard product of Euclidean spaces. We make this choice for consistency with the standard formulation above; however, we emphasize that our machinery is generic to the choice of CRDC. This means we could just as easily derive a distributed MTL training algorithm for some other non-standard CRDC. Now, each task’s loss is a smooth map $\mathcal{L}_i^D : \mathbb{R}^{P_i} \times \mathbb{R}^{P_0} \rightarrow \mathbb{R}$, meaning our optimization domain is $(\mathbf{Euc}, \mathbb{R})$. Note that we have avoided some difficulty by treating each loss as an atomic map, as opposed to the composition of a neural network applied to data followed by a loss function. In the latter case, we would need to consider the set of parameters for the network in addition to the set of data points, and handle the subtleties of updating parameters while ignoring gradients w.r.t the input data. See [7] for a thorough discussion of such concerns. Because the loss $\mathcal{L}_i^D$ is computed with respect to the entire data set $\mathcal{D}_i$ for a task, we

have restricted ourselves to optimization methods that perform true gradient descent as opposed to the stochastic or mini-batch variants which are more common in the machine learning literature. We leave an extension of our work to the SGD setting for future efforts.

With the goal of specifying problem (15) as a morphism of $\mathbf{Opt}_{\text{Euc}}^{\mathbb{R}}$, we first consider the following span that relates shared and task specific parameters for some i .

$$\mathbb{R}^{P_0} \xleftarrow{\pi_0} \mathbb{R}^{P_0} \times \mathbb{R}^{P_i} \xrightarrow{\pi_i} \mathbb{R}^{P_i} \quad (17)$$

To promote (17) to a decorated span, we need only to specify an objective function $\mathbb{R}^{P_0} \times \mathbb{R}^{P_i} \rightarrow \mathbb{R}$. With $\mathcal{L}_i^D$ being the obvious choice, we construct a decorated span $F^i := \left(\mathbb{R}^{P_0} \xleftarrow{\pi_0} \mathbb{R}^{P_0} \times \mathbb{R}^{P_i} \xrightarrow{\pi_i} \mathbb{R}^{P_i}, \mathcal{L}_i^D \right)$. Each F^i is a morphism in the hypergraph category $\mathbf{Opt}_{\text{Euc}}^{\mathbb{R}}$, meaning that we may take the monoidal product $\bigotimes_{i=1}^N F^i$. By Cor. 3.2.1, specifically (8), we know that the monoidal product of our task specific decorated spans is as follows.

$$\bigotimes_{i=1}^N F^i = \left(\prod_{i=1}^N \mathbb{R}^{P_0} \xleftarrow{\prod_{i=1}^N \pi_0^i} \prod_{i=1}^N (\mathbb{R}^{P_0} \times \mathbb{R}^{P_i}) \xrightarrow{\prod_{i=1}^N \pi_i^i} \prod_{i=1}^N \mathbb{R}^{P_i}, \varphi(\mathcal{L}_1^D, \dots, \mathcal{L}_N^D) \right) \quad (18)$$

Note that by Thm. 3.2 the decorating objective function is given by the comparison map:

$$\varphi(\mathcal{L}_1^D, \dots, \mathcal{L}_N^D): \prod_{i=1}^N O(\mathbb{R}^{P_0} \times \mathbb{R}^{P_i}) \rightarrow O\left(\prod_{i=1}^N \mathbb{R}^{P_0} \times \mathbb{R}^{P_i}\right), \quad (19)$$

where $\varphi(\mathcal{L}_1^D, \dots, \mathcal{L}_N^D)$ maps the tuple of task specific losses to their sum:

$$(\mathcal{L}_1^D, \dots, \mathcal{L}_N^D) \mapsto \sum_{i=1}^N \mathcal{L}_i^D \circ \pi^i, \quad (20)$$

where π^i denotes the i 'th projection of $\prod_{i=1}^N (\mathbb{R}^{P_0} \times \mathbb{R}^{P_i})$. We now wish to specify that the parameter space $\mathbb{R}^{P_0}$ is shared among all N loss functions. This is accomplished as follows.

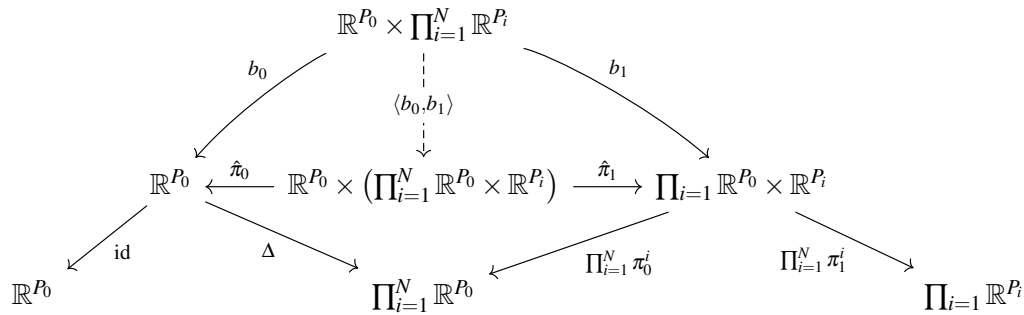
Recall that in a hypergraph category like $\mathbf{Opt}_{\text{Euc}}^{\mathbb{R}}$, we have comultiplication maps

$$\delta_p := \left(\mathbb{R}^p \xleftarrow{\text{id}_p} \mathbb{R}^p \xrightarrow{\Delta} \mathbb{R}^p \times \mathbb{R}^p, 0: \mathbb{R}^p \rightarrow \mathbb{R} \right), \quad (21)$$

where $0: \mathbb{R}^p \rightarrow \mathbb{R}$ is the “empty” decoration, i.e., the constant 0 objective function $0(v) = 0$. The map $\Delta: \mathbb{R}^p \rightarrow \mathbb{R}^p \times \mathbb{R}^p$ is the duplication map $v \mapsto (v, v)$ built-in to any Cartesian category. Thus, to copy the single parameter space $\mathbb{R}^{P_0}$ to all tasks, we simply apply δ_{P_0} N times. We denote this N -fold application $\delta_{P_0}^N$. To recover the problem in (15), all we must do is precompose our monoidal product of task specific objectives with the N -fold copy, yielding the morphism in $\mathbf{Opt}_{\text{Euc}}^{\mathbb{R}}$

$$\begin{aligned} & \left(\left(\bigotimes_{i=1}^N F^i \right) \circ \delta_{P_0}^N, O(\langle b_0, b_1 \rangle) \circ (0 + \sum_{i=1}^N \mathcal{L}_i^D \circ \pi^i) \right) \\ & \left(\left(\bigotimes_{i=1}^N F^i \right) \circ \delta_{P_0}^N, \mathcal{L}^D: (W_0, W_1, \dots, W_N) \mapsto \sum_{i=1}^N \mathcal{L}_i^D(W_0, W_i) \right) \end{aligned} = \quad (22)$$

which can be pictured diagrammatically as:



4.2 Distributed Optimization via Functoriality of Gradient Descent

Algorithm 1: Distributed MTL

```

Input: Initial shared weights  $W_0^0 \in \mathbb{R}^{P_0}$ , and task-specific weights  $W_i^0 \in \mathbb{R}^{P_i}$  for  $i \in \{1, \dots, N\}$ 
Input: A positive real learning rate  $\gamma$ 
 $W_i \leftarrow W_i^0$  for  $i \in \{0, \dots, N\}$ ;
while a stopping criterion is not reached do
    Compute each iteration of the following for-loop in parallel;
    for  $i \leftarrow 1$  to  $N$  do
         $\text{grad\_}W_{0,i} \leftarrow \nabla_{W_0} \mathcal{L}(W_0, W_i)$ ;
         $\text{grad\_}W_i \leftarrow \nabla_{W_i} \mathcal{L}(W_0, W_i)$ ;
         $W_i \leftarrow W_i - \gamma * \text{grad\_}W_i$ ;           /* Update non-shared weights */
    end
     $\text{grad\_}W_0 \leftarrow \sum_{i=1}^N \text{grad\_}W_{0,i}$ ;           /* Gradient of shared weights is sum of
        gradients from each task-specific learner */
     $W_0 \leftarrow W_0 - \gamma * \text{grad\_}W_0$ ;
end
return  $W_0, W_1, \dots, W_N$ 

```

Algorithm 1 describes the distributed scheme for multitask learning derived by applying GD_C to the individual subproblems and composing the resulting dynamical systems. Note that there is value in distributing computation this way as computing the gradients of weights will typically involve non-trivial backpropagation over potentially large networks. With this, we have successfully recovered hard parameter sharing for multitask learning and shown how to make a gradient descent optimizer that can be implemented in a distributed fashion.

5 Discussion and Future Work

Taking stock, it is fair to wonder what has been gained by specifying parameter sharing via decorated spans. One classic argument for the utility of applied category theory is the tight link between graphical problem depiction and its realization in a mathematical or computational data structure. Figure 1 contains two string diagrams which visually capture the structure of multitask learning objectives and optimizers respectively, enabling better communication of models between machine learning practitioners. Moreover, the relative ease with which one can describe compositional objectives supports more complex problem structures. Objectives with traditional regularization terms can be trivially described as compositional objectives, and soft parameter sharing should be fairly easily accommodated via the addition of penalization terms. More exotic hierarchical parameter sharing (see e.g. [17]) should be readily modeled using our approach to MTL. There is a computational benefit too; distributed optimization schemes become easier to implement with a rigorous definition of problem structure. Finally the generality of the techniques discussed allows the parameter sharing pattern of the given morphism to be applied to settings beyond smooth real-valued functions and traditional gradient descent.

As for the general framework defined in Sec. 3, we believe there are interesting connections with previous work on CRDCs. Perhaps the most obvious direction for future work would be to understand the connections with the Para and Lens constructions of [7]. In particular, we hope that a unified treatment of both the compositional structure of models and the compositional structure of objectives can be achieved, perhaps using double categories. Naturally another direction of future work is to find more examples of CRDCs and study not only learning problems in those settings, but other optimization problems such as those arising in operations research.

6 Conclusion

In summary, we have presented hypergraph categories $\mathbf{Opt}_C^R$ and $\mathbf{Dynam}_C$ of open objectives and dynamical systems respectively, and shown that there exists a hypergraph functor between them that maps a generalized objective to its corresponding gradient descent optimizer. Both $\mathbf{Opt}_C^R$ and $\mathbf{Dynam}_C$ are constructed with respect to an underlying Cartesian reverse derivative category, allowing for composition of optimization problems that are defined on a general optimization domain. Such compositionality induces a template for distributed optimization, and provides an accompanying graphical syntax that facilitates communication and sharing. Lastly, to provide evidence that existing machine learning paradigms can be described by our compositional framework, we described parameter sharing models for multitask learning using the hypergraph structure of $\mathbf{Opt}_C^R$, and highlighted the distributed gradient descent algorithm induced by this structure.

7 Acknowledgements

The authors would like to thank Jules Hedges, Simon Willerton, Richard Samuelson, Matthew Hale, Evan Patterson, and David Spivak for helpful conversations. Tyler Hanks was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1842473. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. James Fairbanks was supported by DARPA under award no. HR00112220038 and ONR under award no. N000142312339.

References

- [1] John C. Baez & Blake S. Pollard (2017): *A compositional framework for reaction networks*. *Reviews in Mathematical Physics* 29(09), p. 1750028, doi:10.1142/s0129055x17500283.
- [2] Jonathan Baxter (1997): *A Bayesian/information theoretic model of learning to learn via multiple task sampling*. *Machine learning* 28, pp. 7–39, doi:10.1023/A:1007327622663.
- [3] Christopher M. Bishop (2006): *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- [4] Stephen Boyd (2010): *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. *Foundations and Trends® in Machine Learning* 3(1), pp. 1–122, doi:10.1561/22000000016. Available at <http://www.nowpublishers.com/article/Details/MAL-016>.
- [5] Rich Caruana (1997): *Multitask learning*. *Machine learning* 28, pp. 41–75, doi:10.1023/A:1007379606734.
- [6] Robin Cockett, Geoffrey Cruttwell, Jonathan Gallagher, Jean-Simon Pacaud Lemay, Benjamin MacAdam, Gordon Plotkin & Dorette Pronk (2020): *Reverse Derivative Categories*. In Maribel Fernández & Anca Muscholl, editors: *28th EACSL Annual Conference on Computer Science Logic (CSL 2020), Leibniz International Proceedings in Informatics (LIPIcs)* 152, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 18:1–18:16, doi:10.4230/LIPIcs.CSL.2020.18. Available at <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CSL.2020.18>.
- [7] Geoffrey SH Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2022): *Categorical foundations of gradient-based learning*. In: *European Symposium on Programming*, Springer International Publishing Cham, pp. 1–28, doi:10.1007/978-3-030-99336-8_1.
- [8] Brendan Fong (2015): *Decorated Cospans*. *Theory and Applications of Categories* 30(33), pp. 1096–1120, doi:10.48550/arXiv.1502.00872.
- [9] Brendan Fong (2016): *The algebra of open and interconnected systems*. *arXiv preprint arXiv:1609.05382*, doi:10.48550/arXiv.1609.05382.
- [10] Brendan Fong & David I. Spivak (2019): *Hypergraph categories*. *Journal of Pure and Applied Algebra* 223(11), pp. 4746–4777, doi:10.1016/j.jpaa.2019.02.014. Available at <https://www.sciencedirect.com/science/article/pii/S0022404919300489>.
- [11] Tyler Hanks, Matthew Klawonn, Evan Patterson, Matthew Hale & James Fairbanks (2024): *A Compositional Framework for First-Order Optimization*. *arXiv preprint arXiv:2403.05711*, doi:10.48550/arXiv.2403.05711.
- [12] A. Joyal & R. Street (1993): *Braided Tensor Categories*. *Advances in Mathematics* 102(1), pp. 20–78, doi:10.1006/aima.1993.1055. Available at <https://www.sciencedirect.com/science/article/pii/S0001870883710558>.
- [13] Sophie Libkind, Andrew Baas, Evan Patterson & James Fairbanks (2022): *Operadic Modeling of Dynamical Systems: Mathematics and Computation*. *Electronic Proceedings in Theoretical Computer Science* 372, pp. 192–206, doi:10.4204/EPTCS.372.14. Available at <http://arxiv.org/abs/2105.12282>. ArXiv:2105.12282 [math].
- [14] David E Rumelhart, Geoffrey E Hinton & Ronald J Williams (1986): *Learning representations by back-propagating errors*. *Nature* 323(6088), pp. 533–536, doi:10.1038/323533a0.
- [15] Ozan Sener & Vladlen Koltun (2018): *Multi-task learning as multi-objective optimization*. *Advances in neural information processing systems* 31, doi:10.48550/arXiv.1810.04650.
- [16] Dan Shiebler (2022): *Generalized Optimization: A First Step Towards Category Theoretic Learning Theory*. In Pandian Vasant, Ivan Zelinka & Gerhard-Wilhelm Weber, editors: *Intelligent Computing & Optimization*, Springer International Publishing, Cham, pp. 525–535, doi:10.48550/arXiv.2109.10262.
- [17] Anders Søgaard & Yoav Goldberg (2016): *Deep multi-task learning with low level tasks supervised at lower layers*. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 231–235, doi:10.18653/v1/P16-2038. Available at <https://aclanthology.org/P16-2038>.

- [18] Lijun Zhang, Qizheng Yang, Xiao Liu & Hui Guan (2022): *Rethinking hard-parameter sharing in multi-domain learning*. In: *2022 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, pp. 01–06, doi:10.48550/arXiv.2107.11359.
- [19] Yu Zhang & Qiang Yang (2021): *A survey on multi-task learning*. *IEEE Transactions on Knowledge and Data Engineering* 34(12), pp. 5586–5609, doi:10.48550/arXiv.1707.08114.

A Proofs

The following helper lemma will be crucial going forward.

Lemma A.1. *Let $\phi: X \rightarrow Y$ and $\psi: X' \rightarrow Y'$ be arbitrary morphisms in $\text{Lin}(\mathcal{C})$. Then the following equations hold:*

1. $(\phi \times \psi)^\dagger = \phi^\dagger \times \psi^\dagger$,
2. $\pi_0 \circ (\phi \times \psi) = \phi \circ \pi_0$,
3. $\pi_1 \circ (\phi \times \psi) = \psi \circ \pi_1$.

Proof. For (1), we have

$$\begin{aligned}
 (\phi \times \psi)^\dagger &= \langle \phi \circ \pi_0, \psi \circ \pi_1 \rangle^\dagger \\
 &= (\iota_0 \circ \phi \circ \pi_0 + \iota_1 \circ \psi \circ \pi_1)^\dagger && \text{Lemma 3 in [6]} \\
 &= (\iota_0 \circ \phi \circ \pi_0)^\dagger + (\iota_1 \circ \psi \circ \pi_1)^\dagger && \text{Property of } \dagger\text{-biproducts} \\
 &= \pi_0^\dagger \circ \phi^\dagger \circ \iota_0^\dagger + \pi_1^\dagger \circ \psi^\dagger \circ \iota_1^\dagger && \text{Contravariant functoriality} \\
 &= \iota_0 \circ \phi^\dagger \circ \pi_0 + \iota_1 \circ \psi^\dagger \circ \pi_1 && \text{Lemma 21 in [6]} \\
 &= \langle \phi^\dagger \circ \pi_0, \psi^\dagger \circ \pi_1 \rangle = \phi^\dagger \times \psi^\dagger.
 \end{aligned} \tag{23}$$

For (2), let $(x, x') \in X \times X'$ be generalized elements. Then

$$\pi_0 \circ (\phi \times \psi)(x, x') = \phi(x) = \phi(\pi_0(x, x')). \tag{24}$$

A symmetric argument holds for (3). □

Theorem 3.2. *Given an optimization domain $(\mathcal{C}, R)$, there is a contravariant lax symmetric monoidal functor $O_{\mathcal{C}}^R: (\text{Lin}(\mathcal{C})^{\text{op}}, \times) \rightarrow (\mathbf{Set}, \times)$ defined by the following maps (where we let $O := O_{\mathcal{C}}^R$ for the remainder of the theorem and proof).*

- *Given an object X , $O(X)$ is the set of generalized objectives with domain X , i.e., the hom-set $\mathcal{C}(X, R)$.*
- *Given a linear map $\phi: X \rightarrow Y$ and objective $f: Y \rightarrow R$, $O(\phi)(f): X \rightarrow R$ is the objective $f \circ \phi$ obtained by precomposition.*
- *Given objects X, Y , the product comparison $\varphi_{X,Y}: O(X) \times O(Y) \rightarrow O(X \times Y)$ is defined by*

$$\varphi_{X,Y}(f, g) := f \circ \pi_0 + g \circ \pi_1, \tag{5}$$

where $\pi_0: X \times Y \rightarrow X$ and $\pi_1: X \times Y \rightarrow Y$ are the natural projections.

- *The unit comparison $\varphi_0: \{*\} \rightarrow O(*)$ selects the additive identity 0_R for the hom-set $\mathcal{C}(*, R)$ supplied by the left additive structure.*

Proof. O is plainly functorial as it is just the contravariant hom functor $\mathcal{C}(-, R)$ restricted to act only on the linear maps of $\mathcal{C}$. We still need to verify the symmetric lax monoidal axioms. For naturality of the product comparison, we need the following diagram to commute for all objects X, X', Y, Y' and linear maps $\phi: X' \rightarrow X$ and $\psi: Y' \rightarrow Y$:

$$\begin{array}{ccc} O(X) \times O(Y) & \xrightarrow{O(\phi) \times O(\psi)} & O(X') \times O(Y') \\ \varphi_{X,Y} \downarrow & & \downarrow \varphi_{X',Y'} \\ O(X \times Y) & \xrightarrow{O(\phi \times \psi)} & O(X' \times Y') \end{array}$$

Letting $f: X \rightarrow R$ and $g: Y \rightarrow R$ be arbitrary, following the top path yields the morphism

$$f \circ \phi \circ \pi_0 + g \circ \psi \circ \pi_1 \quad (25)$$

while following the bottom path yields

$$(f \circ \pi_0 + g \circ \pi_1) \circ (\phi \times \psi) = f \circ \pi_0 \circ (\phi \times \psi) + g \circ \pi_1 \circ (\phi \times \psi), \quad (26)$$

where the equality comes from **LA.1**. Now let $x \in X'$ and $y \in Y'$ be generalized elements. Then applying (25) to (x, y) gives $f(\phi(x)) + g(\psi(y))$ while applying (26) to (x, y) gives

$$f(\pi_0(\phi(x), \psi(y))) + g(\pi_1(\phi(x), \psi(y))) = f(\phi(x)) + g(\psi(y)), \quad (27)$$

as desired. Commutativity of the symmetry, associativity, and unitality diagrams all follow from the fact that the hom-set $\mathcal{C}(X' \times Y', R)$ is a commutative monoid. $\square$

Theorem 3.4. *Given a CRDC $\mathcal{C}$, there is a contravariant lax symmetric monoidal functor $D_{\mathcal{C}}: (\text{Lin}(\mathcal{C})^{\text{op}}, \times) \rightarrow (\mathbf{Set}, \times)$ defined by the following maps (where we let $D := D_{\mathcal{C}}$ for the remainder of the theorem and proof).*

- Given an object X , $D(X)$ is the set of generalized dynamical systems with state space X , i.e., the hom-set $\mathcal{C}(X, X)$.
- Given a linear map $\phi: X \rightarrow Y$ and a system $v: Y \rightarrow Y$, $D(\phi)(v)$ is the system $\phi^\dagger \circ v \circ \phi: X \rightarrow X$.
- Given objects X, Y , the product comparison $\varphi_{X,Y}: D(X) \times D(Y) \rightarrow D(X \times Y)$ is defined by

$$\varphi_{X,Y}(v, w) := \pi_0^\dagger \circ v \circ \pi_0 + \pi_1^\dagger \circ w \circ \pi_1. \quad (9)$$

- The unit comparison $\varphi_0: \{*\} \rightarrow D(*)$ is uniquely determined as $\mathcal{C}(*, *)$ is singleton.

Proof. For functoriality of D , let $\phi: X \rightarrow Y$, $\psi: Y \rightarrow Z$, and $v: Z \rightarrow Z$ be arbitrary. For preservation of composition, we have

$$\begin{aligned} D(\phi \circ \psi)(v) &:= (\phi \circ \psi)^\dagger \circ v \circ (\phi \circ \psi) = (\psi^\dagger \circ \phi^\dagger) \circ v \circ (\phi \circ \psi) \\ &= \psi^\dagger \circ (\phi^\dagger \circ v \circ \phi) \circ \psi = (D(\psi) \circ D(\phi))(v), \end{aligned} \quad (28)$$

where the first equality comes from contravariant functoriality of $(-)^{\dagger}$ and the second equality comes from associativity of composition. Similarly, for preservation of identities, we have

$$D(\text{id}_Z)(v) := \text{id}_Z^\dagger \circ v \circ \text{id}_Z = \text{id}_Z \circ v \circ \text{id}_Z = v, \quad (29)$$

where the first equality again comes from functoriality of $(-)^{\dagger}$.

To verify the naturality of the product comparison, we need the following diagram to commute for all $\phi: X' \rightarrow X$ and $\psi: Y' \rightarrow Y$:

$$\begin{array}{ccc} D(X) \times D(Y) & \xrightarrow{D(\phi) \times D(\psi)} & D(X') \times D(Y') \\ \varphi_{X,Y} \downarrow & & \downarrow \varphi_{X',Y'} \\ D(X \times Y) & \xrightarrow{D(\phi \times \psi)} & D(X' \times Y') \end{array}$$

For this, let $(v, w) \in D(X) \times D(Y)$ be arbitrary. Then, following the top path yields

$$\pi_0^{\dagger} \circ \phi^{\dagger} \circ v \circ \phi \circ \pi_0 + \pi_1^{\dagger} \circ \psi^{\dagger} \circ w \circ \psi \circ \pi_1, \quad (30)$$

while following the bottom path yields

$$\begin{aligned} & (\phi \times \psi)^{\dagger} \circ (\pi_0^{\dagger} \circ v \circ \pi_0 + \pi_1^{\dagger} \circ w \circ \pi_1) \circ (\phi \times \psi) \\ &= (\phi \times \psi)^{\dagger} \circ (\pi_0^{\dagger} \circ v \circ \pi_0 \circ (\phi \times \psi) + \pi_1^{\dagger} \circ w \circ \pi_1 \circ (\phi \times \psi)) && \text{Left additivity} \\ &= (\phi \times \psi)^{\dagger} \circ \pi_0^{\dagger} \circ v \circ \pi_0 \circ (\phi \times \psi) + (\phi \times \psi)^{\dagger} \circ \pi_1^{\dagger} \circ w \circ \pi_1 \circ (\phi \times \psi) && \text{Additivity of linear maps} \\ &= (\phi \times \psi)^{\dagger} \circ \pi_0^{\dagger} \circ v \circ \phi \circ \pi_0 + (\phi \times \psi)^{\dagger} \circ \pi_1^{\dagger} \circ w \circ \psi \circ \pi_1 && \text{Lemma A.1} \\ &= (\pi_0 \circ \phi \times \psi)^{\dagger} \circ v \circ \phi \circ \pi_0 + (\pi_1 \circ \phi \times \psi)^{\dagger} \circ w \circ \psi \circ \pi_1 && \text{Contravariant functoriality} \\ &= (\phi \circ \pi_0)^{\dagger} \circ v \circ \phi \circ \pi_0 + (\psi \circ \pi_1)^{\dagger} \circ w \circ \psi \circ \pi_1 && \text{Lemma A.1} \\ &= \pi_0^{\dagger} \circ \phi^{\dagger} \circ v \circ \phi \circ \pi_0 + \pi_1^{\dagger} \circ \psi^{\dagger} \circ w \circ \psi \circ \pi_1, \end{aligned} \quad (31)$$

as desired. The symmetric monoidal coherence axioms again follow from the commutative monoid structure of hom-sets. $\square$

Theorem 3.5. *Given an optimization domain $(\mathcal{C}, R)$, there is a monoidal natural transformation $\text{gd}(\mathcal{C}): O_{\mathcal{C}}^R \Rightarrow D_{\mathcal{C}}$ with components $\text{gd}(\mathcal{C})_X: O_{\mathcal{C}}^R(X) \rightarrow D_{\mathcal{C}}(X)$ defined by the generalized gradient descent optimization scheme, i.e.,*

$$f \mapsto -\mathbf{R}[f]_1. \quad (12)$$

Proof. Let $\text{gd} := \text{gd}_{\mathcal{C}}$, $O := O_{\mathcal{C}}^R$, and $D := D_{\mathcal{C}}$. For naturality, we need to verify that the following diagram commutes for all objects $X, Y \in \text{Lin}(\mathcal{C})$ and linear maps $\phi: Y \rightarrow X$:

$$\begin{array}{ccc} O(X) & \xrightarrow{O(\phi)} & O(Y) \\ \text{gd}_X \downarrow & & \downarrow \text{gd}_Y \\ D(X) & \xrightarrow{D(\phi)} & D(Y) \end{array}$$

Let $f: X \rightarrow R$ be an arbitrary objective. Then, following the top path yields the optimizer

$$-\mathbf{R}[f \circ \phi]_1 := -\mathbf{R}[f \circ \phi] \circ \langle \text{id}_Y, 1_{YR} \rangle. \quad (32)$$

Likewise, following the bottom path yields

$$-\phi^{\dagger} \circ \mathbf{R}[f]_1 \circ \phi := -\phi^{\dagger} \circ \mathbf{R}[f] \circ \langle \text{id}_X, 1_{XR} \rangle \circ \phi. \quad (33)$$

Now consider applying (32) to a generalized element $y \in Y$:

$$\begin{aligned}
 -\mathbf{R}[f \circ \phi] \circ \langle \text{id}_Y, 1_{YR} \rangle(y) &= -\mathbf{R}[f \circ \phi](y, 1_R) \\
 &= -\mathbf{R}[\phi](y, \mathbf{R}[f](\phi(y), 1_R)) \quad \text{Chain rule } \mathbf{RD.5} \\
 &= -\phi^\dagger(\mathbf{R}[f](\phi(y), 1_R)) \quad \text{Linearity.}
 \end{aligned} \tag{34}$$

Finally, applying (33) to the same element gives

$$-\phi^\dagger(\mathbf{R}[f](\langle \text{id}_X, 1_{XR} \rangle(\phi(y)))) = -\phi^\dagger(\mathbf{R}[f](\phi(y), 1_R)) \tag{35}$$

as desired. To verify the transformation is monoidal, we must show that the following diagrams commute:

$$\begin{array}{ccc}
 O(X) \times O(Y) & \xrightarrow{\text{gd}_X \times \text{gd}_Y} & D(X) \times D(Y) \\
 \varphi_{X,Y} \downarrow & & \downarrow \varphi_{X,Y} \\
 O(X \times Y) & \xrightarrow{\text{gd}_{X \times Y}} & D(X \times Y)
 \end{array}
 \quad
 \begin{array}{ccc}
 * & \xrightarrow{\varphi_0} & O(*) \\
 & \searrow \varphi_0 & \downarrow \text{gd}_* \\
 & & D(*)
 \end{array}$$

Let $g: Y \rightarrow R$ be another arbitrary objective. Following the top path of the product comparison diagram yields the system

$$-\pi_0^\dagger \circ \mathbf{R}[f]_1 \circ \pi_0 - \pi_1^\dagger \circ \mathbf{R}[g]_1 \circ \pi_1 \tag{36}$$

while following the bottom path gives

$$\begin{aligned}
 -\mathbf{R}[f \circ \pi_0 + g \circ \pi_1]_1 &= -\mathbf{R}[f \circ \pi_0 + g \circ \pi_1] \circ \langle \text{id}_{X \times Y}, 1_{X \times Y, R} \rangle \\
 &= -(\mathbf{R}[f \circ \pi_0] + \mathbf{R}[g \circ \pi_1]) \circ \langle \text{id}_{X \times Y}, 1_{X \times Y, R} \rangle \quad \mathbf{RD.1} \\
 &= -\mathbf{R}[f \circ \pi_0] \circ \langle \text{id}_{X \times Y}, 1_{X \times Y, R} \rangle - \mathbf{R}[g \circ \pi_1] \circ \langle \text{id}_{X \times Y}, 1_{X \times Y, R} \rangle \quad \mathbf{LA.1} \\
 &= -\mathbf{R}[f \circ \pi_0]_1 - \mathbf{R}[g \circ \pi_1]_1.
 \end{aligned} \tag{37}$$

These can be shown equivalent following the same reasoning as for naturality, namely by applying the chain rule. The unit comparison diagram commutes trivially as $D(*)$ is terminal. $\square$

ViCAR: Visualizing Categories with Automated Rewriting in Coq

Bhakti Shah* William Spencer* Laura Zielinski*
Ben Caldwell Adrian Lehmann Robert Rand
University of Chicago
Chicago, USA

Abstract. We present ViCAR, a library for working with monoidal categories in the Coq proof assistant. ViCAR provides definitions for categorical structures that users can instantiate with their own verification projects. Upon verifying relevant coherence conditions, ViCAR gives a set of lemmas and tactics for manipulating categorical structures. We also provide a visualizer that can display any composition and tensor product of morphisms as a string diagram, showing its categorical structure. This enables graphical reasoning and automated rewriting for Coq projects with monoidal structures.

1 Introduction

Just as category theory provides a unifying framework for diverse concepts in mathematics, category theory can be used in formal verification as a basis for generalization. We can abstract away common structural patterns to yield reusable tactics, lemmas, and techniques for proof assistants. Many constructs that appear in program verification resemble symmetric monoidal categories. ViCAR, a library for Visualizing Categories with Automated Rewriting, takes advantage of the shared structure across symmetric monoidal instances to understand and manipulate morphisms in Coq.

ViCAR emerged from VyZX, an effort to verify the ZX-calculus, a graphical reasoning system for quantum programs [21]. The ZX-calculus [7] forms a symmetric monoidal category whose morphisms are the ZX-diagrams which make up the language. ViCAR generalizes the tools developed in VyZX to assist Coq verification for all concrete monoidal categories. ViCAR consists of three parts: typeclasses for categorical structures in Coq, a visualizer to represent morphisms using string diagrams, and a set of tactics for manipulating typeclass instances.

Other popular examples of monoidal categories include the calculus of relations and matrices. The former is the category whose objects are types and morphisms are binary relations. The latter has matrices as its morphisms between vector spaces. The similarities between these examples are initially unclear. Reframing each categorically, however, reveals their shared structure and hints at how the verification of one could help another. To explore and justify this claim, we imported or implemented these instances in Coq then applied ViCAR’s monoidal category framework. We found that for a well-developed project, we could easily instantiate the relevant categorical definitions. The tactics and visualization gained were valuable and helped project-specific proofs by removing proof and cognitive overhead. We discuss these examples in more detail in Section 6.

ViCAR’s key contributions are as follows:

- We define symmetric monoidal categories in Coq, easily instantiable by user-created structures.

*Equal contribution

- We present an automatic morphism visualizer. While working on a Coq proof, the visualizer will parse the monoidal structure of the current proof state, producing an image of its string diagram representation.
- We provide a set of powerful automation tactics that users can access once they have proven the necessary coherence conditions. They include `foliate`, which automatically rewrites diagrams to common useful structures; `assoc_rw`, which performs rewriting modulo associativity; and `cat_simpl`, a simplification tactic for commonly occurring patterns.
- We formalize the coherence theorem for monoidal categories and develop from it a powerful tactic, `monoidal`, which employs proof by reflection.

We describe ViCAR’s contributions in detail, how to use them, and how they fit into VyZX and other projects with categorical structure. We conclude the paper with some discussion of the next steps for ViCAR. Our work is open source and available on GitHub: <https://github.com/inQWIRE/ViCAR>.

2 Background

Formal verification and the Coq proof assistant Formal verification is grounded in the idea that we can mathematically prove that a computer program satisfies a specification. This guarantees that a given piece of software performs as expected on all possible inputs. Formal verification research focuses on developing new and more effective techniques for these kinds of proofs. One approach uses proof assistants, which are software (usually programming languages) that allow users to express and prove various constructs, often mathematical, in code.

Among the most widely-used proof assistants is Coq [9]. It allows for writing definitions in the style of a dependently-typed programming language and proving statements in the style of mathematical proof. One powerful Coq mechanism is custom tactics, written in the Ltac language, which automate repetitive tasks across proofs. Often, Coq libraries offer a set of tactics to abstract away technical details and perform complex actions. ViCAR aims to provide such tactics for users verifying projects which have categorical structure.

Symmetric monoidal categories Following the definition of Selinger [26], a (planar) monoidal category consists of a base category $\mathcal{C}$ equipped with a bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$ called the tensor product. The tensor product is required to be unital and associative, which means there are natural isomorphisms whose

The figure contains two commutative diagrams. The left diagram is a triangle with vertices $(A \otimes 1) \otimes B$ at the top-left, $A \otimes (1 \otimes B)$ at the top-right, and $A \otimes B$ at the bottom. The top edge is labeled $\alpha_{A,1,B}$, the left edge is labeled $\rho_A \otimes \text{id}_B$, and the right edge is labeled $\text{id}_A \otimes \lambda_B$. The right diagram is a pentagon with vertices $((A \otimes B) \otimes C) \otimes D$ at the top-left, $(A \otimes B) \otimes (C \otimes D)$ at the top-right, $A \otimes (B \otimes (C \otimes D))$ at the middle-right, $A \otimes ((B \otimes C) \otimes D)$ at the bottom-right, and $(A \otimes (B \otimes C)) \otimes D$ at the bottom-left. The edges are labeled with natural isomorphisms: $\alpha_{A \otimes B, C, D}$ (top), $\alpha_{A,B,C \otimes D}$ (right), $\text{id}_A \otimes \alpha_{B,C,D}$ (middle-right), $\alpha_{A,B \otimes C, D}$ (bottom), and $\alpha_{A,B,C} \otimes \text{id}_D$ (left).

Figure 1: The triangle identity (left) and the pentagon identity (right)

components are, for objects $A, B, C \in \mathcal{C}$,

$$\begin{aligned}\lambda_A &: 1 \otimes A \xrightarrow{\sim} A, & (\text{left unitor}) \\ \rho_A &: A \otimes 1 \xrightarrow{\sim} A, & (\text{right unitor}) \\ \alpha_{A,B,C} &: (A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C). & (\text{associator})\end{aligned}$$

These natural isomorphisms are required to be coherent, meaning any diagram of a certain formal class made only of associators and unitors must commute (for details, see Selinger [26]). It has been proven that showing commutativity of just two types of diagrams is sufficient to ensure this powerful result [11]. Specifically, the natural isomorphisms of a monoidal category $\mathcal{C}$ are coherent if and only if the diagrams in Figure 1 commute for all $A, B, C, D \in \mathcal{C}$. This result is called the coherence theorem for monoidal categories, which we may also refer to as monoidal coherence [11]. ViCAR includes a formalization of this theorem and uses it to create the `monoidal` tactic, as we explain in Section 5.

A monoidal category can further be braided, meaning that there is a natural isomorphism whose components are, for objects $A, B \in \mathcal{C}$,

$$\beta_{A,B} : A \otimes B \xrightarrow{\sim} B \otimes A. \quad (\text{braiding})$$

Again, we require that two diagrams called the hexagon identities commute, which ensure the coherence of the braiding. Finally, a braided monoidal category is symmetric if, for all $A, B \in \mathcal{C}$,

$$\beta_{A,B} \circ \beta_{B,A} \simeq 1. \quad (\text{symmetry})$$

Essentially, symmetric monoidal categories have an almost-commutative tensor product. We choose to focus on monoidal and symmetric monoidal categories because of their prevalence and natural correspondence to string diagrams [26]. String diagrams are a graphical way to represent morphisms, useful for parsing complex structure. One of our goals is to unify verification, which is almost always text-based, with visual reasoning. String diagrams are the platform for doing so for monoidal categories. We define them concretely in Section 4.

Categories in verification In mathematics and computer science, symmetric monoidal categories are everywhere. Commonly seen examples include the category of sets, the category of finite vector spaces, and the simply-typed lambda calculus. In verification, active projects whose core structure is a symmetric monoidal category include the verification of the ZX-calculus [21] and that of causal separation diagrams [4]. The ZX-calculus is a graphical language for expressing quantum computation, while causal separation diagrams allow us to reason about parallel processes. Both constructs independently satisfy the definitions of a symmetric monoidal category, though this fact is not directly used in their verification.

Despite being in completely different domains, because of their shared structure, the ZX-calculus and causal separation diagram have properties in common that should be exploited to ease their verification. This is the idea that inspired ViCAR and the gap in formal verification that we wanted to address¹. We bridge this gap with our framework for instantiating monoidal categories, our generalized rewriting tactics, and our morphism visualizer. We prioritize automation and ease—we want users to be able to use ViCAR’s features in their own proofs with minimal additional effort.

Other projects attempt to unify categorical reasoning, proof assistants, and visualization. The Chyp proof assistant, for one, allows users to state rewrite rules axiomatically and produces string diagrams to

¹We note that VyZX and causal separation diagrams are implemented in different proof assistants, Coq and Agda, respectively. We chose to use Coq for ViCAR.

```

Class MonoidalCategoryCoherence {C : Type} {cC : Category C}
  {cCh : CategoryCoherence cC} (mC : MonoidalCategory cC) : Type := {
    triangle (A B : C) :
       $\alpha_{A, I, B} \circ (\text{id}_A \otimes \lambda_B)$ 
       $\simeq \rho_A \otimes \text{id}_B$ ;
    pentagon (A B M N : C) :
       $(\alpha_{A, B, M} \otimes \text{id}_N) \circ \alpha_{A, (B \times M), N} \circ (\text{id}_A \otimes \alpha_{B, M, N})$ 
       $\simeq \alpha_{A, (B \times M), M, N} \circ \alpha_{A, B, (M \times N)}$ ;
    (* Remainder omitted *)
  }.

```

Listing 1: The triangle and pentagon identities in the MonoidalCategoryCoherence typeclass.

visualize the rules in action [19]. ViCAR takes the alternative approach of requiring users to prove their structures are instances of predefined category typeclasses. In exchange for this effort, ViCAR proofs can be used within the greater context of the Coq proof assistant and augment existing Coq projects. ViCAR’s approach allows us to use categorical reasoning to verify existing software, while Chyp is able to more easily handle rewrites modulo associativity.

3 Constructively defining categories in Coq

There are several preexisting examples of implementing category theory in Coq [12, 16]. Though many of these libraries have significant developments, we found they did not align with all of our goals. For instance, we want to separate structural definitions from coherence conditions to enable easy visualization of category instances, independent of their semantics. Moreover, when working with categories in practice, some notion of morphism equivalence is assumed, so we benefit by making this explicit in formal verification. For ViCAR, we are interested in category theory as a means to generalize shared structure. Our library is to be instantiated by active verification projects across a range of domains. We made a number of technical decisions to reflect this purpose.

We implement our categorical definitions using a hierarchy of Coq typeclasses [27], a mechanism similar to interfaces in object-oriented programming. Typeclasses specify and label a collection of types, possibly dependent on each other, and instances of that typeclass must provide a concrete term for each type. For example, Listing 1 gives the part of the typeclass for monoidal categories which translates the coherence conditions from Figure 1. Typeclasses can also inherit from each other. ViCAR’s typeclass hierarchy starts with the base category, then monoidal category, braided monoidal, and finally symmetric monoidal. A project may instantiate as many typeclasses as is suitable for its purposes.

A benefit of using typeclasses is Coq’s inference mechanism, which automatically searches for typeclass instances [27]. This avoids having to reference a particular instance every time one of its terms is used. For example, suppose an instance `catC` of our category typeclass has been declared whose objects have type `C`. Then, if `A, B` and `M` are terms of type `C` and `f` and `g` have types $A \rightsquigarrow B$ and $B \rightsquigarrow M$, the expression `f ∘ g` would type check without the user having to explicitly point to the particular instance. Coq would automatically retrieve it, determining that morphism composition `∘` should be taken as defined by `catC`.

One important way ViCAR differs from existing formalizations of category theory is its separation of structural definitions from coherence conditions. Each abstract category has a typeclass containing just the

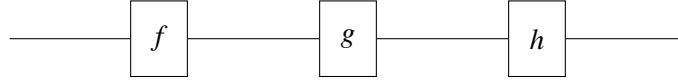
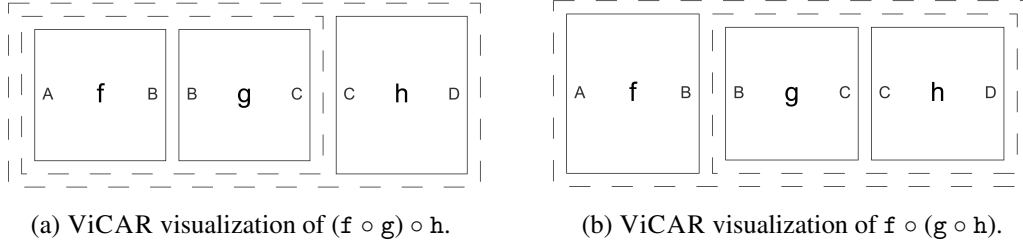
Figure 2: Visualization of both $f \circ (g \circ h)$ and $(f \circ g) \circ h$ in traditional string diagrams.

Figure 3: How ViCAR renders parentheses.

necessary structures, such as identity or the associator, and a different typeclass specifying its coherence conditions. There are two separate hierarchies accordingly. The benefit is that users can access the visualizer without having to prove coherence, which may be more demanding. Instantiating a structural typeclass with existing definitions is enough to begin using our visualizer. Of course, without proving coherence, one has no guarantee that the provided structure actually satisfies the criteria of an abstract category, and therefore cannot use our automation.

Another way ViCAR diverges from other formalizations is by requiring users to supply an explicit equivalence relation for morphisms, denoted $\simeq$, instead of always using built-in equality. Using such an equivalence relation is necessary for many standard constructions because Coq does not support performing a quotient by an equivalence relation. By working over the supplied morphism equivalence, we maintain this flexibility that many implementations rely on. We also chose to use a diagrammatic compose for the notation $\circ$, as our project focuses on visualization.

4 Visualization

Categorical String Diagrams Reasoning about morphisms in categories is assisted by the use of string diagrams, an associated graphical language. String diagrams visually represent monoidal categories that are ordinarily represented in text form. Their notation focuses on morphisms rather than objects, and can therefore very concisely represent complicated expressions. We visualize our categories as string diagrams to reduce cognitive overhead during proof.

Traditional string diagrams omit details such as associativity, so both $f \circ (g \circ h)$ and $(f \circ g) \circ h$ are visualized as Figure 2, for example. This is useful for pen-and-paper proofs but not for proof assistants—formal proofs require each “obvious” detail to be addressed. In Coq, $f \circ (g \circ h)$ and $(f \circ g) \circ h$ are two distinct terms, equivalent only via rewriting (applying a theorem which states they are equal). Identifying necessary rewrites is essential to the proof engineering workflow, so we tweaked ViCAR’s string diagram notation to maintain all parenthesizing explicitly. We represent these parentheses by circumscribing boxes, and the differences can be easily identified as in Figure 3. Color can optionally be added for clarity.

Our visualizations, while convenient for formal proof, are more verbose and add layers of complexity over the simpler diagrams. Future extensions of ViCAR hope to solve this problem, by automatically handling structure. This would allow for diagrams to be canonically represented and for proof engineers to

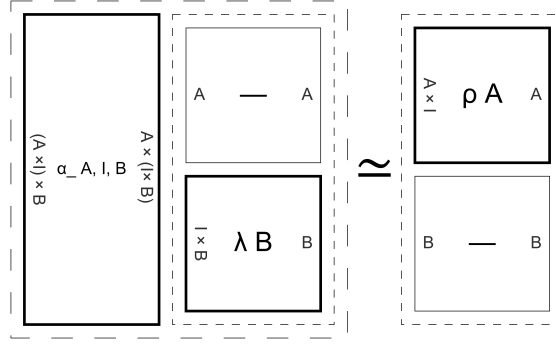


Figure 4: A visualization of the triangle identity.

no longer focus on structural rewrites. While work has been done on rewriting in Coq modulo associativity and commutativity, we found none of it sufficient for what ViCAR needs [3]. The current gap between existing modulo associativity rules and this project relates to rules that depend on the interactions between two operators, like $(f \circ g) \otimes (h \circ t) \simeq (f \otimes h) \circ (g \otimes t)$. There are promising directions being developed on top of e-graph based equality saturation, as we discuss in Section 8.

Visualization semantics and workflow ViCAR allows for visualization of morphisms and morphism equivalences over base category instances up to symmetric monoidal ones. We unlock functionality as we deal with more expressive categories.

- A morphism $f : A \rightsquigarrow B$ is visualized as a quadrilateral, marked with A on the left and B on the right.
- The identity morphism for A is denoted by a wire (horizontal line) with A annotating the input and output positions.
- Morphisms are composed by placing them side-by-side horizontally.
- In monoidal categories, morphisms are tensored by placing them side-by-side vertically.
- Taking the inverse of the morphism is rendered in a box attached to left of the morphism.
- An isomorphism is a morphism with an emphasized bounding box.
- Category-specific terms, such as the associator, are identified as such (with the notation from Section 2).
- In braided monoidal categories, the braiding is rendered as a big cross.

An example which uses several of these features can be seen in Figure 4, which displays the statement of the triangle identity from Listing 1.

One of ViCAR’s most convenient features is its integration into the Coq proof-writing workflow. ViCAR is connected to the coq-lsp VSCode extension [10], which type checks dynamically and prints the current goal state (hypotheses and statements yet to be proved) based on cursor position within a proof. The visualizer renders a string-diagrammatic goal state, alongside the printed one, and updates automatically as the goal changes. The user’s setup can be seen in Figure 5.

Because of this integration, the visualizer is able to use real-time hypothesis information to inform rendering choices. This is useful for distinguishing overloaded notation or opaque variables. For example,

if the variable f in a proof state is a morphism from A to B , the visualizer will make this clear while the goal may not show it explicitly. Association and the effect of braiding may also be unclear from the proof state. The visualizer consults the hypothesis to check the type of each morphism and uses this information to label the inputs and outputs of each morphism’s box.

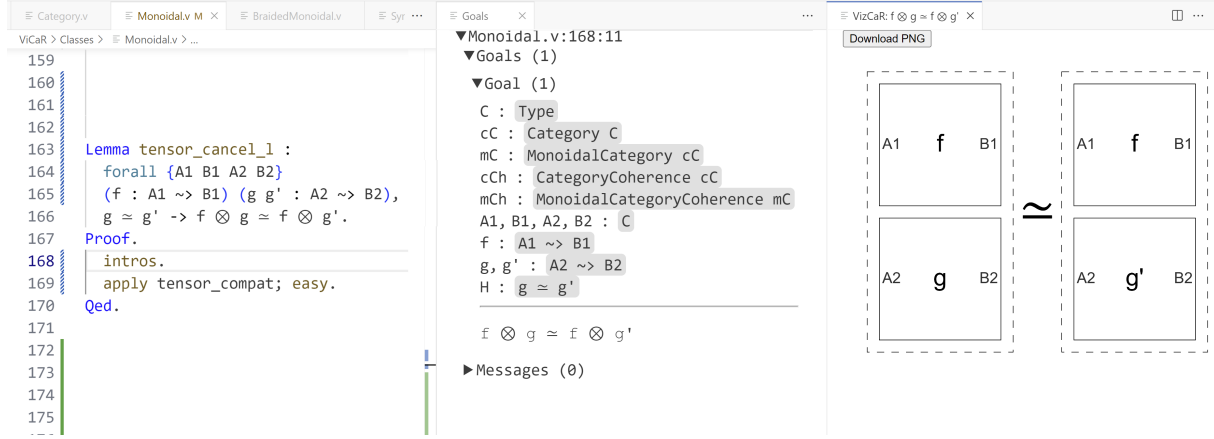


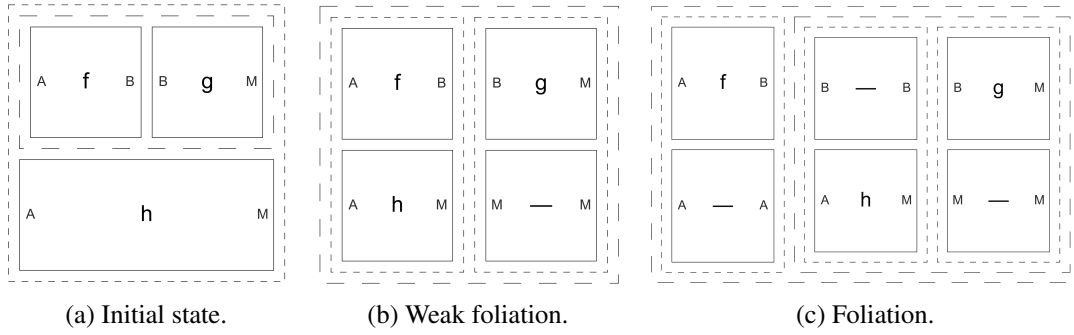
Figure 5: User’s proof-writing IDE state

5 Automation

Categorical structure gives a well-defined domain on which we can develop partial proof automation. Specifically, we can automate proofs of statements that two morphisms are equivalent. This is the form of many theorems in verification projects with monoidal structure. For example, theorems about matrices take the form that certain matrices are equal, and theorems about the ZX-calculus take the form that diagrams are proportional. These types of proofs generally consist of rewrites and share common techniques, such as manipulation of the underlying categorical structure. In this section, we explain ViCAR’s tactics and how they can help with these proofs.

Coercing to categories Coq will often fail to apply general categorical lemmas to proofs about specific typeclass instances because it cannot recognize which terms correspond to categorical structures. For example, suppose the category of matrices (defined in full in Section 6.3) has been declared as a typeclass instance, with composition and identity given by matrix multiplication and the identity matrix. It would not be possible to rewrite the term $I_n \times A$ with the lemma whose statement is $\text{id}_A \circ f \simeq f$. Coq fails to recognize that $\times$ is the composition of a declared category instance and I_n is the identity.

To address this gap, we provide the tactic `categorify`. It performs setup necessary for our automation to identify the structure of the goal. For the category of matrices, calling `categorify` would replace each instance $A \times B$ of matrix multiplication with categorical composition, $A \circ B$, and the same for all other terms used to define the instance. In practice, when using ViCAR, this serves as a setup tactic. At the beginning of the proof, `categorify` is called, which unlocks all the functionality of the visualizer and automation, provided the relevant structural and coherence instances have been defined. This tactic is purely aesthetic, in that it does not change the underlying goal, only the way in which that goal is presented. This change in presentation allows tactics to recognize categorical structures that are implicit.

Figure 6: Visualization of foliation tactics on $(f \circ g) \otimes h$.

Foliation A common representation of a morphism is a foliation [8]. A foliation is a composition of “stacks,” each of which is the tensor of identity morphisms and a single non-identity morphism. Such a representation exists for any diagram and can be considered a standard form and useful for proofs [8]. Formally proving this result for categorical instances, however, requires the complex notion that some morphisms are atomic with respect to this decomposition. For example, in the category of matrices, it is hard to define when a matrix should be decomposable with respect to matrix multiplication and Kronecker product (the tensor product). Our tactic `foliate` computes a foliation of a given morphism from any monoidal categorical instance, proves they are equivalent, and replaces the morphism with its foliation. Often, a full foliation is undesirable, and a more concise form is preferable. The `weak_foliate` tactic performs a partial foliation that allows multiple non-identity morphisms in a stack, but still ensures no stack contains a composition. An example of `weak_foliate` and `foliate` is given in Figure 6 on $(f \circ g) \otimes h$. Its partial foliation is $f \otimes h \circ g \otimes \text{id}_M$ and its full foliation is $f \otimes \text{id}_A \circ (\text{id}_B \otimes h \circ g \otimes \text{id}_M)$.

Associativity: partnering and rewriting A recurring task in many Coq projects is dealing with associativity. Coq expressions have explicit association, so to rewrite a subterm of a sequence of compositions, it is often necessary to perform several rewrites using associativity rules. Even the expression $g \circ f \circ f^{-1}$ cannot be rewritten to g directly, requiring first a rewrite using the associativity condition. In larger expressions, the reassociations which must be performed can be quite laborious to specify manually. We provide several tactics to automate this work.

First, we define the `partner` tactic, which takes two terms as arguments and attempts to reassociate the goal to make these terms syntactically adjacent. After calling `partner f g`, we get $(f \circ g)$ as a subterm of the goal, assuming it can be isolated by reassociation.

Building on this technique, we also define the `assoc_rw` tactic, which takes a lemma as an argument and attempts to reassociate the goal such that the lemma can be rewritten. This works for any lemma whose conclusion, possibly quantified over arguments, has the form $F \simeq g$, where F is some sequence of compositions and g is any morphism. This tactic allows a user to entirely ignore the association of the goal, and simply rewrite according to the sequence of morphisms present in the goal. For example, given a lemma $f \circ g \simeq h$, `assoc_rw` would rewrite its occurrence in the expression $i \otimes (e \circ f \circ g)$, even though this requires reassociation within the argument to the tensor product.

These tactics together entirely obviate the need for manual association of the goal. This delivers on one part of the motivation for using string diagram representations of morphisms: such representations implicitly encode the coherence conditions of monoidal categories by means of topological irrelevance [26].

```

Lemma assoc_rw_example {A B M N : Cobj}
  (f : A ~> B) (g : M ~> N) :
  ( $\beta_{A, M}^{-1} \circ f \otimes g \circ \beta_{B, N}$ 
    $\simeq g \otimes f$ ).
Proof.
  assoc_rw braiding_natural. (* apply braiding lemma *)
  cancel_isos. (* cancellation *)
  reflexivity.
Qed.

```

Listing 2: ViCAR’s tactics being used to easily solve an example lemma.

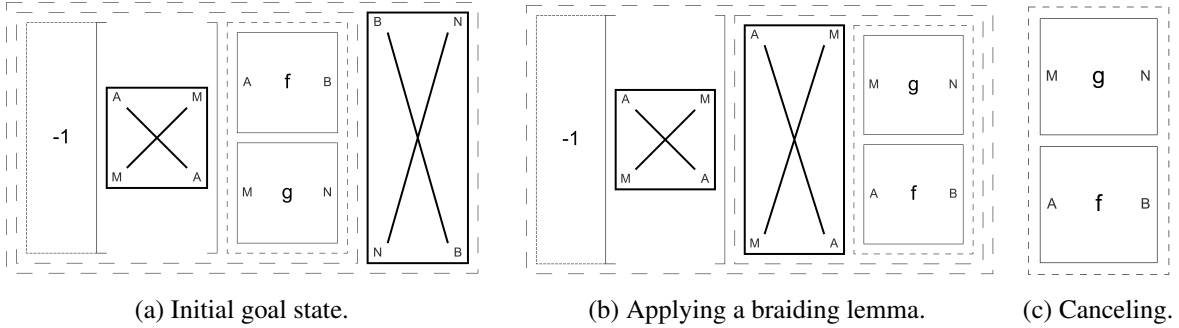


Figure 7: Visualization at each step of Listing 2.

While full topological irrelevance is hard both to define and to prove within Coq, suppressing the consideration of association is a first step in this direction.

Simplifying the goal state Building on these tactics, we provide a number of tactics to simplify the goal state. The tactic `cancel_isos` will cancel any isomorphism adjacent to its inverse, independent of associativity. This is particularly useful for eliminating structural morphisms that accumulate by applications of naturality properties. The tactic `cat_simpl` combines this cancellation with the removal of identity morphisms. These tactics give a quick way to perform common simplifications of a goal without having to consider association. The tactic `cat_easy` attempts to trivial goals that are fundamentally structural. It will cancel isomorphisms and identities, right-associate the goal, and perform weak foliation until the goal is solvable by reflexivity.

Listing 2 and Figure 7 give an example of these tactics in action to almost completely automate a proof. The `assoc_rw` tactic automatically associates the goal to apply the given lemma, a braiding coherence condition. `cancel_isos` then reassociates and cancels out the braiding and its inverse. These tactics can be applied to any proof for a concrete instance, after first calling `categorify`.

Monoidal coherence We formalize the coherence theorem for monoidal categories² in ViCAR and use it to create a tactic, `monoidal`, for solving morphism equivalences that follow directly from monoidal

²Our proof follows exactly the method of [1], to which we refer for details. As in their proof, our result applies only to types with the property of Uniqueness of Identity Proofs. It is not obvious how to overcome this requirement for this proof or for the proof of MacLane, but in practical cases, such as for natural numbers, this property is provable. Moreover, this property is immediate in classical logic, and in particular for types with decidable equality.

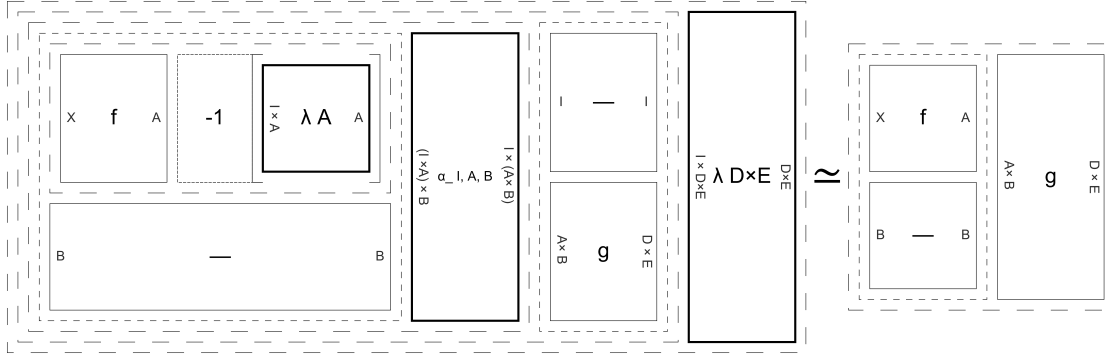


Figure 8: A morphism equivalence lemma which can be immediately proved with one invocation of `monoidal`.

structure. It therefore implements an approximation to traditional string diagram reasoning. This tactic uses proof by reflection, meaning it transforms the goal into an equivalent one provable by pure computation.

The tactic first converts both sides of the goal into inductively-defined intermediate representations. It then applies a theorem that states that two such inductive diagrams are equivalent if their images under a certain function are the same. That function converts the diagrams to a permissive form which mimics traditional string diagrams, meaning without any information about associativity of composition or the tensor product. Finally, the function performs simplification by removing structural isomorphisms, sliding non-identity morphisms as far left through identity morphisms as possible, then removing any extraneous identity morphisms. If the resulting forms are identical, the tactic proves the original morphisms are equivalent. Figure 8 demonstrates the ability of `monoidal` to easily solve certain goals.

6 Example uses of ViCAR

6.1 VyZX

The ZX-calculus is a graphical language to represent quantum operations [7]. Each ZX-diagram represents a linear transformation between qubits and consists of red and green nodes connected by wires. ZX's appeal is that its rewrite rules are easily visualized as graph manipulations. ZX-diagrams can be visually transformed into any equivalent diagram using a finite set of rewrites. The ZX-calculus has been used for simulation [18], circuit optimization [17], and fault tolerance [2] work. For a deeper introduction to the ZX-calculus, refer to one of these surveys [30, 6].

VyZX (Verifying the ZX-Calculus) is an effort to formalize the ZX-calculus in Coq [21]. It gives inductive definitions for diagrams and interprets them through standard matrix semantics for the ZX-calculus. The ZX-calculus corresponds to a symmetric monoidal category whose objects are natural numbers (representing the number of input and output wires) and whose morphisms are ZX-diagrams. Horizontal composition of morphisms (connecting the input and output wires of two diagrams) is associative. The tensor product is given by vertically stacking diagrams, corresponding to addition on natural numbers with identity object 0. The morphism identity given for a natural number n is n wires with no nodes, since composition with this object does not affect any diagram.

As VyZX is a well-developed verification library, most of these categorical structures and necessary

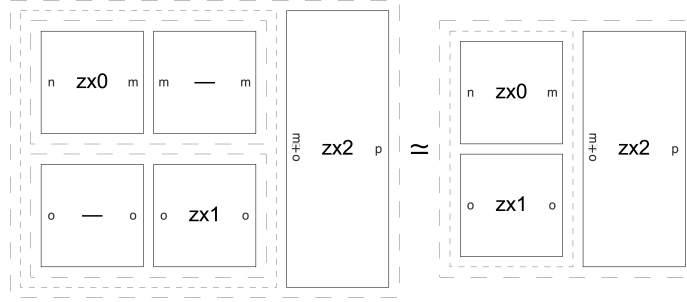


Figure 9: Visualization of the statement $zx0 \leftrightarrow n\_wire\ m \uparrow (n\_wire\ o \leftrightarrow zx1) \leftrightarrow zx2 \propto zx0 \downarrow zx1 \leftrightarrow zx2$.

lemmas already existed. As a result, the coherence conditions for these morphisms were straightforward to prove. The braiding was easily implemented, but proving naturality was a significant task. Fortunately, this task was made easier by proving naturality of braiding for matrices, as discussed in 6.3.

The VyZX visualizer, ZXVIZ, displays ZX-diagrams by parsing an expression into its building blocks and generating a string diagram. We modified ZXVIZ to work for general category instances, abstracting the specific display style used for ZX-diagrams and significantly reworking the foundation of the visualizer to work in the context of a general category. Figure 9 gives an example of ViCAR’s visualizer for a ZX-specific lemma, where $zx0$, $zx1$, and $zx2$ are ZX-diagrams. Its string diagram makes clear this lemma is really just a structural result.

ViCAR grew out of VyZX in an attempt to separate rules specific to the ZX-calculus from rules that are common to any symmetric monoidal category. Typical ZX-calculus literature does not reason about such structure but rather about ZX-diagrams as graphs. Therefore, future VyZX development can benefit from ViCAR’s automation by being able to better ignore structural manipulation of ZX-diagrams.

6.2 Calculus of Relations

In contrast to the previous implementation, the calculus of relations is not a pre-existing Coq project. Its implementation follows the book *Picturing Quantum Processes* [8]. To adapt this work to Coq, we define the objects of our category to be types, and we define morphisms between types T and S to have type $T \rightarrow S \rightarrow \text{Prop}$. As this is defined over any T and S , it can be easily applied to any specific example. By defining base relations such as `sibling` and `parent` over a type `person`, we can construct and visualize relations such as `uncle` in Figure 10. While relations are a toy example, they do demonstrate some insights.

Our implementation of relations was made with ViCAR in mind, and so all proofs are intended for our typeclasses. However, categorical structure is far from a complete description of the properties of relations. For example, it is natural to want to describe the transitive closure of a relation $R : T \rightarrow T \rightarrow \text{Prop}$. There is no clear way to do this solely using categorical construction. Instead, we are able to use Coq’s built-in capabilities to describe a very similar construct: given $R : T \rightarrow T \rightarrow \text{Prop}$, we define R^n , the repeated application of R n times. We can then construct the transitive closure diagrammatically as $\rho\_A \circ (\text{id}_A \otimes \text{any\_nat}) \circ R^n$, where `any_nat` is a relation that relates `unit` to any `nat`. By having ViCAR embedded in an existing proof assistant, we gain the ability to easily create new morphisms as needed and can reason about both structure and properties specific to a particular instance, together.

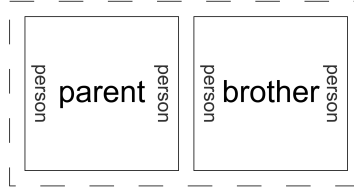


Figure 10: Visualizing the “uncle” relation between two people, $\text{parent} \circ \text{brother}$. (Note that this is diagrammatic composition; parent precedes brother .)

6.3 Matrices

The collection of vector spaces over a field k form the category Vect_k whose morphisms are linear transformations of spaces. This is a monoidal category over the tensor product of vector spaces whose identity element is k . Moreover, this monoidal category is symmetric, as $V \otimes W \cong W \otimes V$.

A commonly used subcategory of Vect_k is the category FinDimVect_k of finite-dimensional vector spaces, which is a symmetric monoidal category in exactly the same way. FinDimVect_k has a skeleton given by vector spaces of the form k^n for $n \in \mathbb{N}$. This skeleton is isomorphic to the category whose objects are natural numbers and whose morphisms from n to m are the $n \times m$ matrices over k , representing linear transformations $k^n \rightarrow k^m$. We implement that category for $k = \mathbb{C}$ using the matrices in the `QuantumLib` library [15], an existing Coq library of verified mathematics for quantum computing. `QuantumLib` is used as the basis of several quantum computing-related projects, including `VyZX`, `SQIR`, `VOQC`, and `QWIRE` [21, 13, 14, 24].

`QuantumLib` already implements most of the definitions and lemmas required to instantiate a category. Matrices in `QuantumLib` are defined as functions $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{C}$. The dimensions of `QuantumLib` matrices are not enforced by the type checker, but instead given as “phantom types” to guide the programmer [25]. This lack of strict dimensions makes the construction and use of the associator and unitors very straightforward.

We implement our category instances using bounded equivalence ($\equiv$) as our morphism equivalence, which says that matrices A, B of dimension $n \times m$ are equivalent if they are equal on all entries within their bounds. The majority of the work in showing matrices are a braided monoidal category involves showing the naturality condition of the braiding. In this category, the braiding is given by the commutation matrices $K_{n,m}$, which have the property that for any $n \times m$ matrix A and $p \times q$ matrix B , we have $K_{p,n}(A \otimes B) = (B \otimes A)K_{q,m}$ [22]. Due to phantom types, the associator and unitor can be defined as the identity matrix interpreted with the appropriate dimensions. In proofs, these matrices can be immediately canceled, making the coherence conditions easy to prove.

A diagrammatic representation of matrix expressions involving the Kronecker encodes facts that textual representations do not. For example, Figure 11 shows the distributivity of the Kronecker product over matrix multiplication. The diagram highlights the structure of the vector spaces on which these matrices act, making a non-obvious equation more believable.

7 Related Work

Category theory has been formalized many times in Coq. Some formalizations such as `Coq-HoTT` [12] and `UniMath` [29] are built on top of homotopy type theory (HoTT) [28]. Other works such as `Coq-Categories` [23] built category theory using typeclasses as we did. However, a commonality between

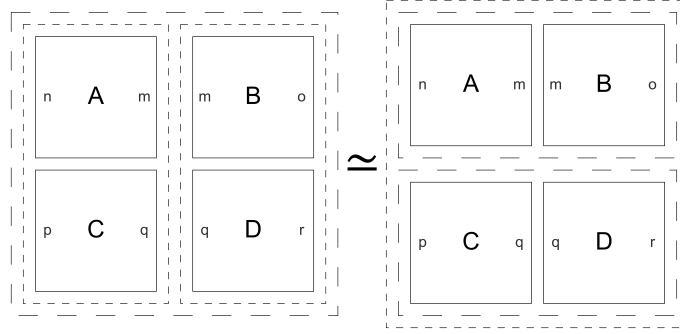


Figure 11: Visualization of the Kronecker mixed-product, $(A \otimes C) \times (B \otimes D) \equiv (A \times B) \otimes (C \times D)$. The variables B and C appear swapped in the text, but the diagram reveals that only the associativity changes.

all the projects we observed is that their goal is to verify large category-theoretic results, and different libraries focus on different parts of category theory [12]. Ultimately, we choose to create our own library to minimize dependencies, such as on HoTT, and allow ourselves flexibility to change parts of the library as needed, since we uniquely focus on automation and applications.

Category theory leans on visual intuition, so it should be no surprise that visualization has played a central role in formalizing categories. Many visualizers display commutative diagrams, rather than string diagrams, and several of these projects have robust interactive editors, such as YADE [20]. Other projects such as Chyp [19] focus on specifying theories and visualizing abstract string diagrams, rather than being applicable to other projects. Another visualizer, commutative-diagrams [5], adds tactics and interaction for manipulating string diagrams in Coq, though it does not use associativity information as we do. Our visualizer lacks interaction, but all the interactive visualizers we found have some dependency on UniMath, which replaces some of Coq’s core libraries and can cause compatibility issues.

8 Future Directions

ViCAR provides a framework for visualization and automated rewriting by defining typeclasses for categorical and monoidal structure. The language of string diagrams can further describe rigid symmetric monoidal categories, with the unit and counit of the category depicted as half turns [26]. ViCAR can be extended with this structure, allowing it to capture a wide variety of process theories.

Many symmetric monoidal categories are interpreted using a semantics function. In fact, the semantics are often used to define morphism equivalence. We aim to provide infrastructure within ViCAR to allow translation between different symmetric monoidal categories for semantic interpretation. This would greatly simplify constructs in categories equipped with a ViCAR-based semantics by lifting them.

Seeing the advantage that the `assoc_rw` tactic gives us, we want to further expand the capabilities of rewriting structure. In fact, we would like to completely be able to rewrite without worrying about structure. To achieve this goal, we are working on an e-graph equality saturation-based solver for different structural configurations. The solver will work by ingesting Coq statements into a custom AST, and then the e-graph solver will use the structural rewrite rules in ViCAR. Once a proof is found, it will be exported into Coq and checked. To build this proof, one must translate dependent type information into an e-graph system. We propose handling by translating SMC structure without dependent types and then rebuilding dependent types as functions within the e-graph. With this system, we will have all the guarantees we

need, as rules can check the dependent type reconstruction function output, given that the diagram given to the solver is valid. This will further facilitate practical reasoning about categorical structures in Coq.

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award numbers FA95502310361 and FA95502310406, and EPiQC, an NSF Expedition in Computing, under Grant No. CCF-1730449.

References

- [1] Ilya Beylin & Peter Dybjer (1996): *Extracting a proof of coherence for monoidal categories from a proof of normalization for monoids*. In Stefano Berardi & Mario Coppo, editors: *Types for Proofs and Programs*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 47–61, doi:10.1007/3-540-61780-9_61.
- [2] Hector Bombin, Daniel Litinski, Naomi Nickerson, Fernando Pastawski & Sam Roberts (2023): *Unifying flavors of fault tolerance with the ZX calculus*, doi:10.22331/q-2024-06-18-1379. arXiv:2303.08829.
- [3] Thomas Braibant & Damien Pous (2011): *Tactics for reasoning modulo AC in Coq*. In: *International Conference on Certified Programs and Proofs*, Springer, pp. 167–182, doi:10.1007/978-3-642-25379-9_14.
- [4] Jonathan Castello, Patrick Redmond & Lindsey Kuper (2023): *Inductive diagrams for causal reasoning*. arXiv:2307.10484.
- [5] Luc Chabassier & Bruno Barras (2023): *A graphical interface for diagrammatic proofs in proof assistants*. In: *29th International Conference on Types for Proofs and Programs TYPES 2023—Abstracts*, p. 98.
- [6] Bob Coecke (2023): *Basic ZX-calculus for students and professionals*. arXiv:2303.03163.
- [7] Bob Coecke & Ross Duncan (2008): *Interacting Quantum Observables*. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfssdóttir & Igor Walukiewicz, editors: *Automata, Languages and Programming*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 298–310, doi:10.1007/978-3-540-70583-3_25.
- [8] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, doi:10.1017/9781316219317.
- [9] The Coq Development Team (2012): *The Coq Reference Manual, version 8.4*. Available electronically at <http://coq.inria.fr/doc>.
- [10] The Coq LSP Developers (2023): *GitHub - ejgallego/coq-lsp: Visual Studio Code Extension and Language Server Protocol for Coq* — [github.com](https://github.com/ejgallego/coq-lsp). <https://github.com/ejgallego/coq-lsp>.
- [11] Samuel Eilenberg & G Max Kelly (1966): *Closed categories*. In: *Proceedings of the Conference on Categorical Algebra: La Jolla 1965*, Springer, pp. 421–562, doi:10.1007/978-3-642-99902-4_22.
- [12] Jason Gross, Adam Chlipala & David I Spivak (2014): *Experience implementing a performant category-theory library in Coq*. In: *Interactive Theorem Proving: 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings 5*, Springer, pp. 275–291, doi:10.1007/978-3-319-08970-6_18.
- [13] Kesha Hietala, Robert Rand, Shih-Han Hung, Liyi Li & Michael Hicks (2021): *Proving Quantum Programs Correct*. In Liron Cohen & Cezary Kaliszyk, editors: *12th International Conference on Interactive Theorem Proving (ITP 2021), Leibniz International Proceedings in Informatics (LIPIcs)* 193, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 21:1–21:19, doi:10.4230/LIPIcs.ITP.2021.21.
- [14] Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu & Michael Hicks (2021): *A Verified Optimizer for Quantum Circuits*. *Proc. ACM Program. Lang.* 5(POPL), doi:10.1145/3434318.

- [15] INQWIRE Developers (2022): *INQWIRE QuantumLib*. Available at <https://github.com/inQWIRE/QuantumLib>.
- [16] John Wiegley (2022): *Category Theory in Coq*. Available at <https://github.com/jwiegley/category-theory>.
- [17] Aleks Kissinger & John van de Wetering (2020): *PyZX: Large Scale Automated Diagrammatic Reasoning*. In Bob Coecke & Matthew Leifer, editors: *Proceedings 16th International Conference on Quantum Physics and Logic*, Chapman University, Orange, CA, USA., 10-14 June 2019, *Electronic Proceedings in Theoretical Computer Science* 318, Open Publishing Association, pp. 229–241, doi:10.4204/EPTCS.318.14.
- [18] Aleks Kissinger & John van de Wetering (2022): *Simulating quantum circuits with ZX-calculus reduced stabiliser decompositions*. *Quantum Science and Technology* 7(4), p. 044001, doi:10.1088/2058-9565/ac5d20.
- [19] Alex Kissinger (2023): *GitHub - akissinger/chyp: An interactive theorem prover for string diagrams — github.com*. <https://github.com/akissinger/chyp>.
- [20] Ambroise Lafont (2024): *A diagram editor to mechanise categorical proofs*. In: *35es Journées Francophones des Langages Applicatifs (JFLA 2024)*.
- [21] Adrian Lehmann, Ben Caldwell, Bhakti Shah & Robert Rand (2023): *VyZX: Formal Verification of a Graphical Quantum Language*. arXiv:2311.11571.
- [22] Jan R. Magnus & H. Neudecker (1979): *The Commutation Matrix: Some Properties and Applications*. *The Annals of Statistics* 7(2), pp. 381–394. Available at <http://www.jstor.org/stable/2958818>.
- [23] Adam Megacz (2007): *A coinductive monad for prop-bounded recursion*. In: *Proceedings of the 2007 workshop on Programming languages meets program verification*, pp. 11–20, doi:10.1145/1292597.1292601.
- [24] Jennifer Paykin, Robert Rand & Steve Zdancewic (2017): *QWIRE: A Core Language for Quantum Circuits*. In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL '17*, Association for Computing Machinery, New York, NY, USA, pp. 846–858, doi:10.1145/3009837.3009894. Available at https://jpaykin.github.io/papers/prz_qwire_2017.pdf.
- [25] Robert Rand, Jennifer Paykin & Steve Zdancewic (2018): *Phantom Types for Quantum Programs*. Available at <https://popl18.sigplan.org/event/coqpl-2018-phantom-types-for-quantum-programs>. Talk at The Fourth International Workshop on Coq for Programming Languages (CoqPL '18).
- [26] P. Selinger (2010): *A Survey of Graphical Languages for Monoidal Categories*. In: *New Structures for Physics*, Springer Berlin Heidelberg, pp. 289–355, doi:10.1007/978-3-642-12821-9_4.
- [27] Matthieu Sozeau & Nicolas Oury (2008): *First-Class Type Classes*. In: *Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics, TPHOLs '08*, Springer-Verlag, Berlin, Heidelberg, p. 278–293, doi:10.1007/978-3-540-71067-7_23.
- [28] The Univalent Foundations Program (2013): *Homotopy Type Theory: Univalent Foundations of Mathematics*. <https://homotopytypetheory.org/book>, Institute for Advanced Study.
- [29] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson et al.: *UniMath — a computer-checked library of univalent mathematics*. available at <http://unimath.org>, doi:10.5281/zenodo.10849216. Available at <https://github.com/UniMath/UniMath>.
- [30] John van de Wetering (2020): *ZX-calculus for the working quantum computer scientist*, doi:10.48550/arXiv.2012.13966. arXiv:2012.13966.

A Coalgebraic Model of Quantum Bisimulation

Lorenzo Ceragioli

IMT School for Advanced Studies, Lucca, Italy

lorenzo.ceragioli@imtlucca.it

Elena Di Lavore

University of Pisa, Pisa, Italy

elena.dilavore@di.unipi.it

Giuseppe Lomurno

University of Pisa, Pisa, Italy

giuseppe.lomurno@phd.unipi.it

Gabriele Tedeschi

University of Pisa, Pisa, Italy

gabriele.tedeschi@phd.unipi.it

Recent works have shown that defining a behavioural equivalence that matches the observational properties of a quantum-capable, concurrent, non-deterministic system is a surprisingly difficult task. We explore coalgebras over distributions taking weights from a generic effect algebra, which subsumes probabilities and *quantum effects*, a physical formalism that represents the probabilistic behaviour of an open quantum system. To abide by the properties of quantum theory, we introduce monads graded on a partial commutative monoid, intuitively allowing composition of two processes only if they use different quantum resources, as prescribed by the *no-cloning theorem*. We investigate the relation between an open quantum system and its probabilistic counterparts obtained when instantiating the input with a specific quantum state. We consider Aczel-Mendler and kernel bisimilarities, advocating for the latter as it characterizes quantum systems that exhibit the same probabilistic behaviour for all input states. Finally, we propose operators on quantum effect labelled transition systems, paving the way for a process calculi semantics that is parametric over the quantum input.

1 Introduction

The recent development of quantum technologies calls for grounded methods for modelling and verifying computing systems that exploit quantum phenomena like superposition and entanglement. In particular, concurrent processes are of main interest, like communication protocols [34, 3, 30, 17] and distributed implementations of algorithms via the *Quantum Internet* [7, 38]. Such systems are essentially composed by a set of non-deterministic agents that run in parallels, can communicate, and have access to a quantum memory over which they can perform operations. The simplest quantum memory is the qubit, which takes values from a complex vector space: it can be in one of two standard basis states $|0\rangle$ and $|1\rangle$, as well as in any linear combination of them. Agents can observe the state of a qubit only through a measurement on a chosen basis, a special kind of quantum operation that destroys the qubit and returns a probabilistic classical outcome, which depends on the qubit state. As an additional constraint, the *no-cloning theorem* forbids copying qubits, which thus cannot be shared by multiple agents [33]. Despite the considerable effort [29, 12, 16, 13, 10], there is still no clear notion of behavioural equivalence for such systems, and most of the proposed ones lack a decision procedure.

In [8], the authors have proposed a new semantic model for quantum protocols, namely quantum effect labelled transition systems (qLTSs). These use *quantum effects* as weights, in the same way probabilities are used in a probabilistic labelled transition system (pLTS). Coming from quantum measurement theory, quantum effects represent the observable probabilistic properties that a quantum system expresses when initialized with a given input quantum state. This indeed allows modelling the agents in quantum protocols: they are essentially non-deterministic procedures acting on an initial quantum state, which

is known only when a specific implementation is used in a concrete setting. Roughly, qLTSs model processes that are parametric over their quantum input.

In this paper we build on this concept, characterizing quantum effects and probabilities as effect algebras. We investigate their (sub)distributions, and we give effect labelled transition systems ($\mathbb{E}$ LTSs), a uniform coalgebraic framework encompassing non-deterministic, probabilistic and quantum concurrent systems. The coalgebraic language is well suited to treat dynamical systems in their essential features, and allows us to extend properties and constructs of probabilistic systems to quantum ones. We introduce monads graded on a partial commutative monoid (PCM), allowing us to grade effect distributions over their quantum resources, the copy of which is forbidden by the no-cloning theorem. Transition systems graded over this PCM intuitively represent quantum computations that consume some quantum resources: only computations using disjoint resources can be composed. Besides, thanks to our peculiar grading, we define a commutative Kronecker product of effects, and thus a commutative multiplication of effect distributions, generalizing the joint distribution operator of the probabilistic case [2].

We investigate kernel and Aczel-Mendler (AM) bisimilarities for $\mathbb{E}$ LTSs, and relate them by generalizing previous results. By applying our findings to the quantum setting, we prove that each quantum state ρ defines a functor from qLTSs to pLTSs that “instantiates” a quantum process to the probabilistic behaviour it exhibits when the measured quantum state is ρ . We study which equivalences are preserved and/or reflected by these functors, giving us a categorical way to pinpoint the correct notion of equivalence for quantum systems, i.e. the one that equates all and only the processes that exhibit the same observable probabilistic behaviour. Both AM and kernel bisimilarities correctly relate indistinguishable quantum processes only. However, only the latter is complete and relates all the indistinguishable processes, given that the weights of the qLTS are taken from a finite effect algebra of quantum effects.

Finally, we investigate operators over $\mathbb{E}$ LTSs, paving the way for an $\mathbb{E}$ LTS semantics of quantum process calculi. We provide a generalized, compositional parallel operator which can model different notions of synchronization (CCS, CSP, ACT) and different kinds of “weights” (nondeterministic, probabilistic or quantum systems). In addition, we introduce a purely quantum operator of “partial evaluation” of qLTSs that instantiates the value of some of the qubits. These operators are defined as functors between $\mathbb{E}$ LTSs, thus they preserve bisimilarity. While similar operators in the literature typically act on the final coalgebra, treating them as functors allows for “multi sorted” operators: partial evaluation reduces the resources used by the computation; parallel composition joins them by increasing the number of qubits (only if they are compatible, i.e. of different quantum systems). A qLTS semantics for quantum processes would reduce the problem of deciding probabilistic bisimilarity of two protocols for all possible input quantum states to just checking kernel bisimilarity of two given qLTSs, which can be solved via standard techniques [26].

Related Works. Usually, works from the process algebra literature [29, 14, 12, 13, 10, 9] define the semantics of quantum processes via pLTSs, and strive to adapt probabilistic bisimilarity to capture the peculiar observable properties of quantum values. The defined pLTSs are made of configurations, i.e. pairs of quantum values and syntactic processes, impeding algorithmic verification of processes when the quantum input is not given (the only symbolic approach [15] has been proved too strict [10]). We instead introduce a purely quantum transition system: we do not represent directly quantum values but only their observable probabilistic features in the form of effects.

This work builds on [8]: we use the language of distribution monads, their coalgebras and their coalgebraic bisimilarity to generalize the concepts of quantum distribution, quantum transition systems and Larsen-Skou bisimilarity presented there.

Noticeably, our effect distribution monad subsumes both the probability distribution monad, the quantum effect distributions of [8] and the quantum monad of [1], which is based on projectors, a subset of quantum effects. Our graded monad definition extends the non-graded effect monad presented in [24], which takes weights in effect monoids, and not in the larger class of graded effect monoids.

Some similar notion to our $\mathbb{E}$ LTSs has already been proposed: effect-valued Quantum Markov Chain of [18] uses sequential effect composition instead of tensor product; QLTS of [35] uses superoperators instead of effects, so to capture also non-destructive measurements. The author of [35] introduces also two different notions of bisimilarity, that we recover in our minimal, effect-based setting as AM and kernel bisimilarity. However, none of these works feature nondeterminism, which are needed for modelling protocols, nor they investigate correctness by comparing their findings with the probabilistic behaviour of quantum systems. A final novelty of our work is investigating operators over labelled transition systems, suitable for modelling quantum protocols, e.g. via process calculi.

Synopsis. In section 2 we give some background about effect algebras, quantum computing and quantum effects. In section 3 we present effect distributions and their monads. In section 4 we investigate coalgebras on effect distributions and their bisimilarity, and we discuss $\mathbb{E}$ LTSs and their operators in section 5. Finally, we conclude in section 6, and we give the proof sketches in the appendix.

2 Background

We recall the definitions of partial commutative monoid and effect algebra. Then, we give some background on quantum computing and quantum effects. We refer to [33] for further reading on quantum computing, to [21] for the details of quantum effects, and to [23] for their algebraic treatment.

2.1 Partial Commutative Monoid and Effect Algebra

Partial commutative monoids obey the properties of commutative monoids, but $+$ is not always defined.

Definition 1. A *partial commutative monoid (PCM)* is a tuple $\langle M, 0, + \rangle$ (often referred as M) with $0 \in M$ and $+: M \times M \rightarrow M$ a partial binary operation on M such that for all $a, b, c \in M$ the following hold:

- (Commutativity) $a \perp b$ implies $b \perp a$ and $a + b = b + a$;
- (Associativity) $b \perp c$ and $a \perp (b + c)$ implies $a \perp b$ and $(a + b) \perp c$ and also $(a + b) + c = a + (b + c)$;
- (Zero) $0 \perp a$ and $0 + a = a$.

Here, $a \perp b$ means that a and b are orthogonal, i.e. $a + b$ is defined. A PCM homomorphism is a function $f: M \rightarrow N$ on the underlying carrier sets such that $f(0) = 0$, and $a \perp b$ implies $f(a) \perp f(b)$ and $f(a + b) = f(a) + f(b)$. PCMs and their homomorphisms form the category **PCM**.

Every PCM induces a preorder on the carrier set M where $a \preceq b$ if and only if $\exists c \in M. a + c = b$.

Effect algebras are a special kind of PCMs for which an inverse operation is defined.

Definition 2. An *effect algebra* [23] is a tuple $\langle \mathbb{E}, 0, +, \cdot' \rangle$ (often referred as $\mathbb{E}$) with $\langle \mathbb{E}, 0, + \rangle$ a PCM and $\cdot': \mathbb{E} \rightarrow \mathbb{E}$ a unary operation such that, for all $e \in \mathbb{E}$:

- $e' \in \mathbb{E}$ is the unique element in $\mathbb{E}$ such that $e + e' = 1$ with $1 = 0'$;
- $e \perp 1$ implies $e = 0$.

An *effect homomorphism* is a PCM homomorphism that also preserves $\cdot'$. The category **EA** of effect algebras and effect homomorphisms is the full subcategory of **PCM** whose objects are effect algebras.

Effect algebras have a partial order $\sqsubseteq$ (defined as $\preceq$) and a partial operation $e_1 - e_2$ returning the unique e_3 such that $e_2 + e_3 = e_1$. **EA** is a symmetric monoidal category with $2 = \{0, 1\}$ its unit object. There is a bijective correspondence between morphisms $\mathbb{E}_A \otimes \mathbb{E}_B \rightarrow \mathbb{E}_C$ and bihomomorphisms $\mathbb{E}_A \times \mathbb{E}_B \rightarrow \mathbb{E}_C$ (a bihomomorphism is such that the morphisms obtained by fixing either object are homomorphisms).

An example of effect algebras are probabilities: real numbers in the interval $[0, 1]$ with $+$ the arithmetic sum and e' defined as $1 - e$. Probabilities allow for defining *probability (sub)distributions* over a given set S , i.e. functions $\Delta : S \rightarrow [0, 1]$ such that $\sum_{s \in S} \Delta(s) \leq 1$. For each $s \in S$, we let $\bar{s}$ be the *point distribution* that assigns 1 to s . Given a finite set of non-negatives reals $\{p_i\}_{i \in I}$ such that $\sum_{i \in I} p_i \leq 1$, the weighted sum $\sum_{i \in I} p_i \cdot \Delta_i$ defines a distribution such that $(\sum_{i \in I} p_i \cdot \Delta_i)(s) = \sum_{i \in I} p_i \Delta_i(s)$.

Probability distributions form a *convex set* [6], meaning that for all distributions Δ, Θ and for all real $p \in [0, 1]$ there exists a distribution $\Delta_p \oplus \Theta$ defined as $p \cdot \Delta_1 + (1 - p) \cdot \Delta_2$. Given a function f between convex sets X and Y , we call f convex if it preserves the $_p \oplus$ operator, i.e. if $f(x_1)_p \oplus f(x_2) = f(x_1)_p \oplus f(x_2)$. We denote as **Conv**(X, Y) the set of convex functions between X and Y .

2.2 Quantum Computing

A (finite-dimensional) *Hilbert space*, denoted as $\mathcal{H}$, is a complex vector space equipped with a binary operator $\langle \cdot | \cdot \rangle : \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{C}$ called *inner product*, defined as $\langle \psi | \phi \rangle = \sum_i \alpha_i^* \beta_i$, where $|\psi\rangle = (\alpha_1, \dots, \alpha_i)^T$ and $|\phi\rangle = (\beta_1, \dots, \beta_i)^T$, with T the transpose. We indicate column vectors as $|\psi\rangle$ and their conjugate transpose as $\langle \psi| = |\psi\rangle^\dagger$. The state of an isolated physical system is represented as a *unit vector* $|\psi\rangle$ (called *state vector*), i.e. a vector such that $\langle \psi | \psi \rangle = 1$. The two-dimensional Hilbert space $\mathbb{C}^2$ is called a *qubit*. The vectors $\{|0\rangle = (1, 0)^T, |1\rangle = (0, 1)^T\}$ form an orthonormal basis of $\mathbb{C}^2$, called the *computational basis*. Other important vectors in $\mathbb{C}^2$ are $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, which form the *Hadamard basis*. Intuitively, different bases represent different observable properties of a quantum system. Note that $|+\rangle$ and $|-\rangle$ are non-trivial linear combinations of $|0\rangle$ and $|1\rangle$, roughly meaning that the property associated with the computational basis is undetermined in $|+\rangle$ and $|-\rangle$. In the quantum jargon, $|+\rangle$ and $|-\rangle$ are *superpositions* with respect to the computational basis. Symmetrically, $|0\rangle$ and $|1\rangle$ are superpositions with respect to the Hadamard one.

We represent the state space of a composite physical system as the *tensor product* of the state spaces of its components. Let $\mathcal{H}$ and $\mathcal{H}'$ be n and m -dimensional Hilbert spaces: their tensor product $\mathcal{H} \otimes \mathcal{H}'$ is an $n \cdot m$ Hilbert space. Moreover, if $\{|\psi_1\rangle, \dots, |\psi_n\rangle\}$ and $\{|\phi_1\rangle, \dots, |\phi_m\rangle\}$ are bases of respectively $\mathcal{H}$ and $\mathcal{H}'$, then $\{|\psi_i\rangle \otimes |\phi_j\rangle \mid i = 1, \dots, n, j = 1, \dots, m\}$ is a basis of $\mathcal{H} \otimes \mathcal{H}'$, where $|\psi\rangle \otimes |\phi\rangle$ is the Kronecker product. We often omit the tensor product and write $|\psi\rangle |\phi\rangle$ or $|\psi\phi\rangle$. Note that such product is not commutative. Categorically, finite-dimensional Hilbert spaces with the Kronecker product and the conjugate transpose form the dagger compact category *FDHilb*. Further references are available in [11].

The density operator formalism puts together quantum systems and probability by considering mixed states, i.e. *probability distributions of quantum states*. A point distribution $|\overline{\psi}\rangle$ (called a pure state) is represented by the matrix $|\psi\rangle\langle\psi|$. In general, a probability distribution Δ of n -dimensional states is represented as the matrix $\rho \in \mathbb{C}^{n \times n}$, known as its *density operator*, with $\rho = \sum_i \Delta(\psi_i) |\psi_i\rangle\langle\psi_i|$. Recall that a complex matrix N is called *positive semi-definite*, shortly *positive*, when $\langle \psi | N | \psi \rangle \geq 0$ for all $|\psi\rangle$. The *Löwner order* is the partial order defined by $L \sqsubseteq L'$ whenever $L' - L$ is positive. Given a d -dimensional Hilbert space $\mathcal{H}$, density operators coincide with the positive matrices in $\mathbb{C}^{d \times d}$ of trace one, we denote them as $DM_{\mathcal{H}} = \{\rho \in \mathbb{C}^{d \times d} \mid \rho \sqsupseteq 0_d, \text{tr}(\rho) = 1\}$. Density operators form a convex set, where the convex combination operator is defined by $\rho_p \oplus \sigma = p\rho + (1 - p)\sigma$.

Density operators can describe the state of a subsystem of a composite quantum system. Let $\mathcal{H}_S$

denote the Hilbert space of a physical system S , then $\mathcal{H}_{S_1} \otimes \mathcal{H}_{S_2}$ is the Hilbert space of a composite system with subsystems S_1 and S_2 . Given a (not necessarily separable) $\rho \in \mathcal{H}_{S_1} \otimes \mathcal{H}_{S_2}$, the *reduced density operator* of system S_1 , $\rho_1 = \text{tr}_{S_2}(\rho)$, describes the state of S_1 , with tr_{S_2} the *partial trace over S_2* , defined as the linear transformation such that $\text{tr}_{S_2}(|\psi\rangle\langle\psi'| \otimes |\phi\rangle\langle\phi'|) = |\psi\rangle\langle\psi'| \text{tr}(|\phi\rangle\langle\phi'|)$ for each $|\psi\rangle\langle\psi'| \in DM_{\mathcal{H}_{S_1}}$ and $|\phi\rangle\langle\phi'| \in DM_{\mathcal{H}_{S_2}}$.

2.3 Quantum Effects

Quantum measurements are needed for describing systems that exchange information with the environment. Performing a measurement on a quantum state returns a probabilistic classical result and either destroys or otherwise changes the quantum system. We focus in this paper on destructive measurements.

The simplest kind of measurements are *quantum effects* (simply called effects in quantum textbooks [21]), i.e. yes-no tests over quantum systems. Each effect can be represented as a positive matrix smaller than the identity in the Löwner order. We denote the set of effects on a d -dimensional Hilbert space $\mathcal{H}$ as $\mathbb{Q}_{\mathcal{H}} = \{L \in \mathbb{C}^{d \times d} \mid 0_d \sqsubseteq L \sqsubseteq \mathbb{I}_d\}$, where $\mathbb{I}_d$ is the $d \times d$ identity matrix. The probability of getting a “yes” outcome when measuring an effect L on a state $|\psi\rangle$ is given by the *Born rule* $\text{tr}(L\rho)$. Effects of dimension d form an effect algebra with the matrix sum, $\mathbb{I}_d$ as 1 and $L' = \mathbb{I}_d - L$. Furthermore, the induced partial order $\sqsubseteq$ is exactly the Löwner order.

A measurement with n outcomes is a set $\{L_1, \dots, L_n\}$ of effects, such that the *completeness* equation $\sum_{i=1}^n L_i = \mathbb{I}$ holds. If the state of the system is ρ before the measurement, then the probability of the i outcome occurring is $p_i = \text{tr}(L_i\rho)$. As examples of measurements, consider M_{01} and $M_{\pm}$ that project a state into the elements of the computational and Hadamard basis of $\mathbb{C}^2$ respectively, with M_{01} defined as $\{|0\rangle\langle 0|, |1\rangle\langle 1|\}$ and $M_{\pm}$ as $\{|+\rangle\langle +|, |-\rangle\langle -|\}$. Applying the measurement M_{01} on $|0\rangle$ returns the outcome associated with $|0\rangle\langle 0|$ with probability 1. When measuring $|+\rangle$, instead, the same result occurs with probability $\frac{1}{2}$. Notice that a measurement for a composite system may measure only some of the qubits, e.g. $\{|0\rangle\langle 0| \otimes \mathbb{I}, |1\rangle\langle 1| \otimes \mathbb{I}\}$ measures (in the computational basis) the first qubit of a pair.

Density operators and effects are dual, as effects are isomorphic to the convex functions from the set of density operators to the probability interval. The isomorphism is given by the Born rule.

Theorem 1. *It holds that $\mathbb{Q}_{\mathcal{H}} \cong \mathbf{Conv}(DM_{\mathcal{H}}, [0, 1])$ through the isomorphism $L \mapsto \lambda\rho. \text{tr}(L\rho)$ [21].*

Roughly, effects can be considered as probabilities *parametrized* on an unknown quantum state.

3 Effect Distributions

Here we lay some fundamental definitions about effect distributions, which generalize probability distributions by using elements of a generic effect algebra $\mathbb{E}$ as weights, instead of the usual $[0, 1]$ interval. We show that the features of effect distributions descend from the algebraic structure of their weights: if the latter have a (commutative) multiplication, then effect distribution form a (commutative) monad; if we have a morphism of weights, we get a natural transformations of distributions. We exemplify these concepts on probability distributions and on quantum effect distributions. Finally, we show that, like probability distributions, quantum distributions do form a commutative monad, albeit a graded one.

Following the work of [23], for each effect algebra $\mathbb{E}$ we build a functor of effect distributions.

Definition 3 (Effect Distributions). *Given an effect algebra $\langle \mathbb{E}, 0, +, \cdot' \rangle$ we define the functor of effect (sub)distributions $D_{\mathbb{E}} : \mathbf{Set} \rightarrow \mathbf{Set}$ by*

$$D_{\mathbb{E}}X = \left\{ \Delta \in \mathbb{E}^X \mid \text{supp}(\Delta) \text{ is finite, } \sum_{x \in \text{supp}(\Delta)} \Delta(x) \sqsubseteq 1_{\mathbb{E}} \right\} \quad (D_{\mathbb{E}}f)(\Delta) = \lambda y \in Y. \sum_{x \in f^{-1}(y)} \Delta(x)$$

where $\text{supp}(\Delta)$ is the set $\{x \in X \mid \Delta(x) \neq 0\}$, and $\sum$ is the n -ary sum in $\mathbb{E}$.

Our running examples will be sub-probability distributions $D_{[0,1]}$, featuring weights the unit interval, and quantum distributions over some Hilbert space $\mathcal{H}$, associated to the effect algebra $\mathbb{Q}_{\mathcal{H}}$. Notice how, if $\mathcal{H}$ is 1 dimensional, the effect algebra $\mathbb{Q}_{\mathcal{H}}$ is isomorphic to $[0,1]$, and $D_{\mathbb{Q}_{\mathcal{H}}}$ to $D_{[0,1]}$. We will often write an effect distribution $\Delta = \{x_i \mapsto e_i\}_{x_i \in X}$ in a compact form, as $\Delta = \sum_{x_i \in X} e_i \bullet x_i$. Thus, some examples of distributions on $X = \{a, b\}$ could be

$$\frac{1}{2} \bullet s + \frac{1}{2} \bullet t \in D_{[0,1]}X \cong D_{\mathbb{Q}_{\mathbb{C}}}X \quad |0\rangle\langle 0| \bullet s + |1\rangle\langle 1| \bullet t \in D_{\mathbb{Q}_{\mathbb{C}^2}}.$$

Intuitively, the distribution on the left models a random choice, where we toss a fair coin to decide between s and t . The one on the right is a quantum choice, i.e. a measurement, where we choose by measuring a qubit in the computational basis and by taking s or t according to the result.

Transformation of weights gives us transformations of distributions. Formally, we have the following theorem, rephrasing Proposition 21 of [23].

Theorem 2. *Each effect morphism $m : \mathbb{E} \rightarrow \mathbb{F}$ yields a natural transformation $m \circ \cdot : D_{\mathbb{E}} \Rightarrow D_{\mathbb{F}}$. Moreover, if m is injective then the components of $m \circ \cdot$ are injective.*

For example, for each $\rho \in DM_{\mathcal{H}}$, there is an effect algebra homomorphism m_{ρ} from $\mathbb{Q}_{\mathcal{H}}$ to $[0,1]$ given by $m_{\rho}(L) = \text{tr}(L\rho)$. Intuitively, this homomorphism applies an input density operator ρ that transforms each effect L to the probability of observing the "yes" outcome when performing the L measurement on a quantum system in state ρ . From this we get a transformation from $\Delta \in D_{\mathbb{Q}_{\mathcal{H}}}X$ to $m_{\rho} \circ \Delta \in D_{[0,1]}X$. Different ρ yields different transformations: given $\Delta = |0\rangle\langle 0| \bullet s + |1\rangle\langle 1| \bullet t$, we have

$$m_{|+\rangle\langle +|} \circ \Delta = \frac{1}{2} \bullet s + \frac{1}{2} \bullet t \quad m_{|1\rangle\langle 1|} \circ \Delta = 1 \bullet t.$$

It is well known that, when weights have a multiplicative structure, weighted distributions form a monad [24, 31, 5]. While probabilities do have a multiplication, forming a monoid in $\mathbf{EA}$ (called an effect monoid in [24]), quantum effects do not.

Notice how two effects $L_1, L_2 \in \mathbb{Q}_{\mathcal{H}}$ can be multiplied via the tensor product, but the resulting effect $L_1 \otimes L_2$ is in $\mathbb{Q}_{\mathcal{H} \otimes \mathcal{H}}$. Essentially, the carrier of $\mathbb{Q}_{\mathcal{H}}$ is not closed for the tensor operation. We thus introduce *graded effect monoids*, allowing quantum effects to be tensored together, using grades to keep track of the resulting Hilbert space.

Instead of grading on a monoid or a monoidal category, we use a partial commutative monoid (PCM) as grades. As we will see, this is useful for ensuring linearity of quantum resources.

Definition 4 (Graded Effect Monoid). *Given a partial commutative monoid $\langle M, 0, + \rangle$, an M -graded effect monoid is a graded monoid object in $\mathbf{EA}$, that is*

- for each $m \in M$, an effect algebra $\mathbb{E}_m$
- for each $m \perp n \in M$, an effect morphism $\nabla_{m,n} : \mathbb{E}_m \otimes \mathbb{E}_n \rightarrow \mathbb{E}_{m+n}$
- an effect morphism $\eta : 2 \rightarrow \mathbb{E}_0$

such that the two following diagrams on the left commute

$$\begin{array}{ccc} (\mathbb{E}_m \otimes \mathbb{E}_n) \otimes \mathbb{E}_o & \xrightarrow{\alpha} & \mathbb{E}_m \otimes (\mathbb{E}_n \otimes \mathbb{E}_o) \\ \nabla_{m,n} \otimes 1 \downarrow & & \downarrow 1 \otimes \nabla_{n,o} \\ \mathbb{E}_{m+n} \otimes \mathbb{E}_o & & \mathbb{E}_m \otimes \mathbb{E}_{n+o} \\ \nabla_{m+n,o} \downarrow & \swarrow \nabla_{m,n+o} & \\ \mathbb{E}_{m+n+o} & & \end{array} \quad \begin{array}{ccc} 2 \otimes \mathbb{E}_m & \xrightarrow{\eta \otimes id} & \mathbb{E}_0 \otimes \mathbb{E}_m \\ \lambda \downarrow & \swarrow \nabla_{0,m} & \\ \mathbb{E}_m & & \\ \rho \uparrow & \swarrow \nabla_{m,0} & \\ \mathbb{E}_m \otimes 2 & \xrightarrow{id \otimes \eta} & \mathbb{E}_m \otimes \mathbb{E}_0 \end{array} \quad \begin{array}{ccc} \mathbb{E}_m \otimes \mathbb{E}_n & \xrightarrow{B_{\mathbb{E}_m, \mathbb{E}_n}} & \mathbb{E}_n \otimes \mathbb{E}_m \\ \nabla_{m,n} \downarrow & \swarrow \nabla_{n,m} & \\ \mathbb{E}_{m+n} & & \end{array}$$

If also the diagram on the right commutes, where B is the braiding natural transformation, then we say that the graded effect monoid is commutative.

Each monoid object can be considered as being graded on the degenerate 1-element monoid $\{0\}$, thus all effect monoids described in [24] are an example of trivially graded effect monoids.

For the quantum case, we grade our effect monoid over the PCM of disjoint sets of quantum systems. Note that it is indeed impossible to perform multiple measurements on the same quantum state, because the no-cloning theorem forbids copying quantum data, and measurements destroy the measured system. For complying with this physical constraint, our grading keeps track of the qubits measured by the effects, and the partial sum reflects that the composition of effects is defined over disjoint systems only.

Definition 5 (PCM of Quantum Systems). Assume a finite set of quantum systems $\text{Sys} = \{S_i\}$, each associated with the Hilbert space $\mathcal{H}_{S_i}$ of finite dimension d_i . Let $\mathcal{S} = \langle \mathcal{P}(\text{Sys}), \emptyset, \uplus \rangle$ be the PCM where $\mathcal{P}(\text{Sys})$ is the powerset of Sys and $\uplus$ is the partial disjoint union, i.e. $C_i \uplus C_j$ is defined only if $C_i \cap C_j = \emptyset$, and in that case $C_i \uplus C_j = C_i \cup C_j$. We associate each collection of systems $C \in \mathcal{P}(\text{Sys})$ with a Hilbert space defined as $\mathcal{H}_\emptyset = \mathbb{C}$ and $\mathcal{H}_C = \bigotimes_{S \in C} \mathcal{H}_S$, where we impose that the arguments of the (non-commutative) Kronecker product are ordered according to their indices.

Assume for example Sys to be the finite set of qubits $\{q_1, \dots, q_n\}$. Then, our collections $C \in \mathcal{S}$ are subsets of these qubits, and the grading allows us to keep track of which of them are consumed by each measurement, forbidding the tensoring of effects that use a common qubit. Indeed, the PCM above grades an effect monoid of quantum effects.

Theorem 3 (Effect Monoid of Quantum Effect). Quantum effects carry a commutative $\mathcal{S}$ -graded effect monoid structure, given by:

- The effect algebra $\mathbb{E}_C$ is $\mathbb{Q}_{\mathcal{H}_C}$ for all collection of systems $C \in \mathcal{P}(\text{Sys})$
- The operator $\nabla_{C,D} : \mathbb{E}_C \otimes \mathbb{E}_D \rightarrow \mathbb{E}_{C \uplus D}$ is denoted as $\boxtimes$ and defined by $L_1 \boxtimes L_2 = \text{Sort}_{C,D}(L_1 \otimes_k L_2)$, where $\otimes_k$ is the Kronecker product between $L_1 \in \mathbb{Q}_{\mathcal{H}_C}$ and $L_2 \in \mathbb{Q}_{\mathcal{H}_D}$, and $\text{Sort}_{C,D}$ is the unitary transformation which “sorts” the Hilbert Space $\mathcal{H}_C \otimes \mathcal{H}_D$ into $\mathcal{H}_{C \uplus D}$.
- The effect morphism $\eta : 2 \rightarrow \mathbb{E}_\emptyset$ defined by $\{0 \mapsto 0, 1 \mapsto 1\}$.

Since quantum effects are only a PCM-graded monoid and not a monoid in **EA**, the resulting distribution functor is a PCM-graded monad and not a monad.

Definition 6 (Graded Monad). Given a partial commutative monoid $\langle M, 0, + \rangle$, an M -graded monad on **Set** is a graded monoid in the category of **Set**-endofunctors, that is

- for each $m \in M$, an endofunctor $T_m : \mathbf{Set} \rightarrow \mathbf{Set}$
- a natural transformation $\eta : \text{Id} \rightarrow T_0$ called unit
- for each $m \perp n \in M$, a natural transformation $\mu_{m,n} : T_m T_n \rightarrow T_{m+n}$ called multiplication

such that the following diagrams commute

$$\begin{array}{ccc}
 T_m T_n T_o & \xrightarrow{T_m \mu_{n,o}} & T_m T_{n+o} \\
 \mu_{m,n} T_o \downarrow & & \downarrow \mu_{m,n+o} \\
 T_{m+n} T_o & \xrightarrow{\mu_{m+n,o}} & T_{m+n+o}
 \end{array}
 \qquad
 \begin{array}{ccc}
 T_m & \xrightarrow{T_m \eta} & T_m T_0 \\
 \eta T_m \downarrow & \searrow & \downarrow \mu_{m,0} \\
 T_0 T_m & \xrightarrow{\mu_{0,m}} & T_m
 \end{array}$$

PCM-graded monads generalize the notion of monoid-graded monad of [27], asking for an associative multiplication only when the grades are orthogonal.

Es expected, whenever $\{\mathbb{E}_m\}$ is a graded effect monoid, $\{D_{\mathbb{E}_m}\}$ has a graded monad structure.

Theorem 4 (Graded Monads of Graded Effect Monoids). *If $\{\mathbb{E}_m\}$ is an M -graded effect monoid, there is a graded monad $\{D_{\mathbb{E}_m}\}$ with unit $\eta : Id \rightarrow D_{\mathbb{E}_0}$ and multiplication $\mu_{m,n} : D_{\mathbb{E}_m} D_{\mathbb{E}_n} \rightarrow D_{\mathbb{E}_{m+n}}$ given by*

$$\eta(x) = 1_{\mathbb{E}_0} \bullet x \quad \mu_{m,n}(\sum_i e_i \bullet \Delta_i)x = \sum_i \nabla_{m,n}(e_i, \Delta_i(x))$$

For brevity, we will write Q_C for $D_{\mathbb{Q}_{\mathcal{H}_C}}$ in the following. Recall that, in the probabilistic case, μ corresponds to the weighted sum of probability distributions [22]. For quantum distributions, instead, multiplication models concatenated quantum measurements, by taking the (sorted) Kronecker product of the effects: given a set of quantum effect distributions Δ_i in $Q_C X$ and a quantum distribution Θ in $Q_{C'}(Q_C X)$ associating each distribution Δ_i with a quantum effect, the multiplication returns a distribution in $Q_{C' \oplus C} X$ associating each $x \in X$ with $\sum_i \Theta(\Delta_i) \boxtimes \Delta_i(x)$ (if C and C' are disjoint). This coincides with the intuition of measuring first the qubits in C , and then, based on the outcome, performing a second measurement over C' , i.e. Δ_i .

For later use, we finally define commutative graded monads. Commutativity allows us to reduce the pairing of effect distributions to a distribution of pairs, permitting a well-behaved definition of the parallel composition of effect distributions. For the (trivially graded) probabilistic case, this corresponds to the joint probability distribution construction à la [37].

Definition 7 (Commutative Graded Monad). *An M -graded monad $\{T_m\}$ on **Set** is strong if it has left strength $\sigma_{m,X,Y} : X \times T_m Y \rightarrow T_m(X \times Y)$ and a right strength $\tau_{m,X,Y} : T_m X \times Y \rightarrow T_m(X \times Y)$ which respect the monoidal structure $\times$ on **Set** and the graded multiplication and unit of T (the diagrams it must satisfy are just the graded version of the usual ones for strong monads). A strong graded monad is commutative if for all $m \perp n \in M$ there is a canonical natural transformation $\alpha : T_m X \times T_n Y \rightarrow T_{m+n}(X \times Y)$ defined by any of the two compositions $\mu_{m,n} \circ T_m \sigma_n \circ \tau_m = \mu_{n,m} \circ T_n \tau_m \circ \sigma_n$.*

Theorem 5. *If $\{\mathbb{E}_m\}$ is a commutative graded effect monoid, then $\{D_{\mathbb{E}_m}\}$ is a commutative graded monad with its canonical strength.*

4 $D_{\mathbb{E}}$ -Coalgebras for Quantum Systems

We first investigate coalgebras defined on effect distributions and their bisimilarities. Then, we focus on the specific case of quantum systems, discussing the relation between probabilistic and quantum coalgebras and characterizing correctness for a quantum behavioural equivalence.

4.1 $D_{\mathbb{E}}$ -Coalgebras and their Bisimilarities

We start by recalling the definitions of coalgebra and coalgebra homomorphism.

Definition 8 (Coalgebra). *Let $F : \mathbf{Set} \rightarrow \mathbf{Set}$ be an endofunctor on the **Set** category. An F -coalgebra is a pair (X, c) , with X an object of **Set**, and a morphism $c : X \rightarrow FX$ (also written $X \xrightarrow{c} FX$).*

Given two F -coalgebras (X, c) and (Y, d) , A morphism $f : X \rightarrow Y$ is an F -coalgebra homomorphism, written $f : (X, c) \rightarrow (Y, d)$, if $Ff \circ c = d \circ f$. F -coalgebra homomorphisms include the identity and are closed for composition, and thus F -coalgebras constitute a category, denoted as $\mathbf{Set}_F$.

We consider the two most common kinds of bisimilarity on coalgebras: AM-bisimilarity and kernel bisimilarity, of which we recall the definitions below.

Definition 9 (Aczel-Mendler Bisimilarity). *A relation $R \subseteq X \times Y$ is an AM-bisimulation between the F -coalgebras $X \xrightarrow{c} FX$ and $Y \xrightarrow{d} FY$ if there exists an F -coalgebra $R \xrightarrow{e} FR$ such that the projections $\pi_1 : R \rightarrow X$ and $\pi_2 : R \rightarrow Y$ are coalgebra homomorphisms, i.e. the diagram in Figure 1a commutes. Two states $x \in X$ and $y \in Y$ are AM-bisimilar, written $x \sim_{AM} y$, if xRy for some AM-bisimulation R .*

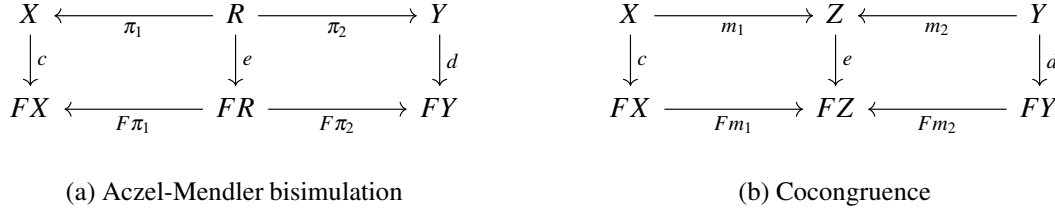


Figure 1: The two commutative diagrams defining AM and Kernel bisimulations, respectively

Definition 10 (Kernel Bisimilarity). A cocongruence between two F -coalgebras $X \xrightarrow{c} FX$ and $Y \xrightarrow{d} FY$ is a cospan $X \xrightarrow{m_1} Z \xleftarrow{m_2} Y$ such that there exists an F -coalgebra $Z \xrightarrow{e} FZ$ making m_1 and m_2 coalgebra homomorphisms, i.e. the diagram in Figure 1b commutes. We call kernel bisimulation the pullback in **Set** of a cocongruence $X \xrightarrow{m_1} Z \xleftarrow{m_2} Y$, that is the relation R such that xRy if and only if $m_1(x) = m_2(y)$. Kernel bisimilarity $\sim_k$ is the largest kernel bisimulation.

In particular, we are interested in the case where the endofunctor F is $D_{\mathbb{E}}$ for some effect algebra $\mathbb{E}$. In this case, the equality of kernel and AM bisimilarities entirely depends on the underlying effect algebra $\mathbb{E}$. In line with the results in [19, 35], the two relations coincide if and only if $\mathbb{E}$ is decomposable.

Definition 11 (Decomposable Effect Algebra). We say that an effect algebra $\mathbb{E}$ is decomposable if for all $a, b, c, d \in \mathbb{E}$ such that $a \perp b$, $c \perp d$ and $a + b = c + d$, there exists $e_{11}, e_{12}, e_{21}, e_{22} \in \mathbb{E}$ such that $a = e_{11} + e_{12}$, $b = e_{21} + e_{22}$, $c = e_{11} + e_{21}$ and $d = e_{12} + e_{22}$.

Theorem 6. Let $\mathbb{E}$ be an effect algebra. Let $X \xrightarrow{c} D_{\mathbb{E}}X$ and $Y \xrightarrow{d} D_{\mathbb{E}}Y$ be two $D_{\mathbb{E}}$ -coalgebras. For all $x \in X$ and $y \in Y$: (i) $x \sim_{AM} y \implies x \sim_k y$, and (ii) $x \sim_k y \implies x \sim_{AM} y$ if and only if $\mathbb{E}$ is decomposable.

Another topic we are interested in is the relation between $D_{\mathbb{E}}$ -coalgebras over possibly different effect algebras, namely $D_{\mathbb{E}}$ -coalgebras and $D_{\mathbb{E}'}$ -coalgebras for a possibly different $\mathbb{E}'$. In particular, we want to know under which conditions a mapping from $D_{\mathbb{E}}$ -coalgebras to $D_{\mathbb{E}'}$ -coalgebras preserves and reflects bisimilarities. Note that natural transformations between the functors that define a coalgebra have the important property of preserving bisimilarities, as demonstrated in [2].

Theorem 7. Let $\alpha : F \Rightarrow G$ be a natural transformation between two functors F and $G : \mathbf{Set} \rightarrow \mathbf{Set}$. The natural transformation α induces a functor, denoted $\alpha \circ \cdot : \mathbf{Set}_F \rightarrow \mathbf{Set}_G$, which maps objects (X, c) to $(X, \alpha_X \circ c)$ and homomorphisms $f : (X, c) \rightarrow (Y, d)$ to homomorphisms $f : (X, \alpha_X \circ c) \rightarrow (Y, \alpha_Y \circ d)$. Furthermore, bisimilarities are preserved by this functor.

It is then easy to assess that effect morphisms define functors that preserve bisimilarity, thanks to Theorem 2. Note however that functors induced by natural transformations do not reflect bisimilarities, in general. In [2] further conditions are identified in order for functors to reflect bisimilarities.

Theorem 8. Let $\alpha : F \Rightarrow G$ be a natural transformation between two functors F and $G : \mathbf{Set} \rightarrow \mathbf{Set}$. If all components α_X of α are injective then the induced functor $\alpha \circ \cdot$ reflects the kernel bisimilarity.

4.2 The Quantum Case

Hereafter, we apply the results above to the specific case of probability and quantum effect distributions and their monads. Recall that, for brevity, we write Q_C for $D_{\mathbb{Q}_{\mathcal{H}_C}}$. Note that coalgebras $(X, c : X \rightarrow D_{\mathbb{E}}X)$ are essentially a generalization of Markov Chains, coinciding with the usual definition but with partial distributions if $\mathbb{E} = [0, 1]$. Q_C -coalgebras can be seen instead as a quantum version of Markov Chains.

We start by noticing that kernel and AM-bisimilarities coincide in the probabilistic case, thanks to Theorem 6 and since $[0, 1]$ is decomposable (as demonstrated implicitly in [32]). On the contrary, the equality does not hold in the quantum case. Thanks to Theorem 6, it suffices to show the following.

Proposition 1. *If the dimension of $\mathcal{H}$ is greater or equal than 2, then $\mathbb{Q}_{\mathcal{H}}$ is not decomposable.*

A simple example of a case in which kernel and AM-bisimilarities disagree follows.

Example 1. Take $X \xrightarrow{c} Q_C$, with $X = \{x_1, x_2, x_3, x_4\}$ and c defined as below.

$$c(x_1) = |0\rangle\langle 0| \bullet x_3 + |1\rangle\langle 1| \bullet x_4 \quad c(x_2) = |+\rangle\langle +| \bullet x_3 + |-\rangle\langle -| \bullet x_4 \quad c(x_3) = c(x_4) = \mathbb{I} \bullet x_3$$

We cannot build an AM-bisimulation such that $x_1 \sim_{AM} x_2$. However, if we take $Z = \{z_1, z_2\}$ and m such that $m(x_1) = m(x_2) = z_1$, $m(x_3) = m(x_4) = z_2$, then $(X, c) \xrightarrow{m} (Z, d) \xleftarrow{m} (X, c)$ is a cocongruence for d defined as $d(z_1) = d(z_2) = \mathbb{I}_2 \bullet z_2$, i.e. $x_1 \sim_k x_2$.

Since kernel and AM-bisimilarities do not agree even on simple cases, we look for some correctness criteria to help us decide which one to use for comparing quantum systems. For defining them, it is useful to recall the effect homomorphism $m_{\rho}(L) = \text{tr}(L\rho)$ that for each $\rho \in DM_{\mathcal{H}}$ maps a quantum effect to a probability, and the induced natural transformation from Q_C to $D_{[0,1]}$ by Theorem 2. Thanks to Theorem 7, a functor $\downarrow_{\rho}$ is defined for each $\rho \in DM_{\mathcal{H}}$ that maps a Q_C -coalgebra in the $D_{[0,1]}$ -coalgebra characterizing the behaviour of the Quantum Markov Chain when the input state is ρ . Indeed, The functor $\downarrow_{\rho}$ updates the weights of a Quantum Markov Chain by computing the Born rule.

Example 2. Take the Q_C -coalgebra (X, c) from Example 1, then the $D_{[0,1]}$ -coalgebra, $\downarrow_{|0\rangle\langle 0|}(X, c)$ is (X, c') with $c'(x_1) = 1 \bullet x_3$, $c'(x_2) = \frac{1}{2} \bullet x_3 + \frac{1}{2} \bullet x_4$ and $c'(x_3) = c'(x_4) = 1 \bullet x_3$.

Since their probabilistic behaviour is the only observable property of quantum systems, our (first) correctness principle for a bisimilarity is that two elements of the Q_C -coalgebras (X, c) and (Y, d) shall be bisimilar if and only if they are indistinguishable in the probabilistic systems obtained by instantiating their quantum input, i.e. in the $D_{[0,1]}$ -coalgebras $\downarrow_{\rho}(X, c)$ and $\downarrow_{\rho}(Y, d)$ for any $\rho \in DM_{\mathcal{H}}$.

Desiderata 1. *Let C be a collection of quantum systems, and let (X, c) and (Y, d) be Q_C -coalgebras with $x \in X$ and $y \in Y$. Then, the two following conditions must coincide*

- x and y are bisimilar in (X, c) and (Y, d)
- x and y are bisimilar in $\downarrow_{\rho}(X, c)$ and $\downarrow_{\rho}(Y, d)$ for all ρ

By Theorem 7, all the functors $\downarrow_{\rho}$ preserve both bisimilarities, hence one side of the implication holds for both of them. However, Theorem 6 and Proposition 1 imply that AM-bisimilarity in $D_{[0,1]}$ is not reflected by $\downarrow_{\rho}$: it does not satisfy the other direction of Desiderata 1. In particular, AM-bisimilarity fails to relate processes that are probabilistically indistinguishable when instantiated with all ρ .

Example 3. Take (X, c) from Example 1, and recall that $x_1 \sim_k x_2$, while $x_1 \not\sim_{AM} x_2$. From the discussion above we know that the former implies that $x_1 \sim_k x_2$ in $\downarrow_{\rho}(X, c)$ for all ρ . Moreover, the latter confirms that AM-bisimilarity does not meet our desiderata.

Regarding the other direction, we obtain that kernel bisimilarity fully satisfies Desiderata 1 when the quantum effects used in (X, c) and (Y, d) are contained in a finite effect algebra $\mathbb{L} \subsetneq \mathbb{Q}_{\mathcal{H}}$. To prove this result, we build on the following technical lemma, restating our Lemma 4 of [8]. It ensures that, when $\mathbb{L}$ is finite, a single density operator exists that distinguishes every pair of distinct effects in $\mathbb{L}$.

Lemma 1. *Let $\mathbb{L}$ be a finite set of quantum effects in $\mathbb{Q}_{\mathcal{H}}$, then a density operator $\hat{\rho} \in DM_{\mathcal{H}}$ exists such that for each $L, L' \in \mathbb{L}$, $\text{tr}(L\hat{\rho}) = \text{tr}(L'\hat{\rho})$ if and only if $L = L'$.*

This means that among all the effect algebra homomorphisms m_ρ , there is at least an injective one $m_{\hat{\rho}}$, which yields a natural transformation with injective components by Theorem 2. Thus, by Theorem 7 the resulting functor $\downarrow_{\hat{\rho}}$ reflects kernel bisimilarity.

Theorem 9. *Let (X, c) and (Y, d) be two $D_{\mathbb{L}}$ -coalgebras, and let $x \in X$ and $y \in Y$. If x and y are kernel bisimilar in $\downarrow_\rho(X, c)$ and $\downarrow_\rho(Y, d)$ for all ρ , then x and y are kernel bisimilar in (X, c) and (Y, d) .*

As previously stated, this result is based on Lemma 1, which is proved only on finite effect algebras. We expect that this result can be extended at least to the countably infinite case. However, the current model is still capable of representing and comparing the behaviour of systems performing a finite set of different destructive measurements, which is often the case in quantum protocols [34, 3, 30, 17] and in the quantum process calculi literature [10, 28, 16, 13].

We investigate now a different correctness principle, called locally parameterized probabilistic bisimilarity (lpp) in [8]. In the previous approach, two quantum systems are equated if they behave the same when given any possible quantum state. Nevertheless, the input is given globally for the whole Markov Chain. This means we assume that an adversary trying to disprove equivalence can only choose the state once at the beginning. In lpp instead, the adversary can give a different quantum state at each step of the two Markov Chains. In the following, we model this feature coalgebraically. Recall that $L \mapsto \lambda \rho. \text{tr}(L\rho) : \mathbb{Q}_{\mathcal{H}} \rightarrow \mathbf{Conv}(DM_{\mathcal{H}}, [0, 1])$ is an isomorphism from quantum effects to convex parameterized probabilities. Indeed, $\mathbf{Conv}(DM_{\mathcal{H}}, [0, 1])$ is itself an effect algebra, that we name f_C for convenience, and $L \mapsto \lambda \rho. \text{tr}(L\rho)$ is an effect algebra isomorphism. We can therefore define D_{f_C} and a natural transformation $\alpha_{lpp} : Q_C \rightarrow D_{f_C}$ transforming a quantum distribution over X into a distribution that associates each element $x \in X$ with a function that given a quantum input returns a probability. Intuitively, while $\downarrow_\rho$ applies a unique initial input density operator that transforms a Quantum Markov Chain into a probabilistic system, $\alpha_{lpp} \circ \cdot$ allow us to change ρ at every step of our bisimilarity.

Desiderata 2. *Let (X, c) and (Y, d) be Q_C -coalgebras, and let $x \in X$ and $y \in Y$. Then, bisimilarity of x and y in (X, c) and (Y, d) , i.e. when effects are used as weights, must hold if and only if x and y are bisimilar in $(\alpha_{lpp} \circ \cdot)(X, c)$ and $(\alpha_{lpp} \circ \cdot)(Y, d)$, i.e. when convex functions are used as weights in place of their isomorphic effects.*

Kernel bisimilarity over Q_C -coalgebras exactly coincides with this notion of locally parameterized probabilistic bisimilarity.

Theorem 10. *Let (X, c) and (Y, d) be Q_C -coalgebras, and let $x \in X$ and $y \in Y$. Then $x \sim_k y$ in (X, c) and (Y, d) holds if and only if $x \sim_k y$ holds in $(\alpha_{lpp} \circ \cdot)(X, c)$ and $(\alpha_{lpp} \circ \cdot)(Y, d)$.*

An interesting corollary is that, when coalgebras are built over a finite quantum effect algebra $\mathbb{L}$, the stronger adversary that can choose a different quantum input at each step is not capable of discriminating more than the weaker one that only chose once at the beginning.

5 Effect Labelled Transition Systems

We can now define a generalization of probabilistic transition systems to effect distributions, for which we extend the properties of the previous section and introduce both a generic parallel composition and a specifically quantum operator of partial evaluation.

Definition 12 ($\mathbb{E}$ -Labelled Transition System). *Given an effect algebra $\mathbb{E}$, a $\mathbb{E}$ -labelled transition system (ELTS) is a coalgebra $X \xrightarrow{c} \mathcal{P}(D_{\mathbb{E}}X)^L$, where X is a set of states, L is a fixed set of labels, $\mathcal{P}$ is the finitary powerset endofunctor and $D_{\mathbb{E}}$ is the $\mathbb{E}$ -distribution endofunctor on **Set**.*

As is typical for process calculi, we will assume that there exist a special symbol $\tau \in L$ and an involutive unary operation $\bar{\cdot}$ for all labels different from τ . Intuitively, τ represents a silent, invisible action, and whenever μ represents a visible action (e.g. outputting a value), $\bar{\mu}$ represents its "dual" action (e.g. receiving that value). Moreover, we will write $x \xrightarrow{\mu} \Delta$ for $\Delta \in c(x)(\mu)$.

In the following, we deal with $D_{[0,1]}$ LTSs and Q_C LTSs (called pLTSs and qLTSs hereafter), i.e. LTSs over distributions of probabilities and of quantum effects in some $\mathcal{H}_C$ respectively. All the results on $D_{\mathbb{E}}$ -coalgebras of the previous section can be easily extended to more complex coalgebras through *whiskering*. For example, we can whisker the natural transformation $\downarrow_\rho: Q_C \rightarrow D_{[0,1]}$ and the finite powerset functor $\mathcal{P}$, obtaining a natural transformation $\mathcal{P}\downarrow_\rho: \mathcal{P}Q_C \rightarrow \mathcal{P}D_{[0,1]}$ and recovering the previous results for systems that transition non-deterministically to effect distributions.

Corollary 1. *Let (X, c) and (Y, d) be qLTSs, with $x \in X$ and $y \in Y$.*

- $x \sim_{AM} y \implies x \sim_k y$ in general for qLTSs;
- $x \sim_k y \not\implies x \sim_{AM} y$ if the dimension of the Hilbert space is at least two;
- for each ρ over the adequate Hilbert space, there is a functor transforming c and d into the pLTSs c_ρ and d_ρ representing the probabilistic behaviour of the system when the quantum state is ρ ;
- if x and y are kernel bisimilar in c and d , then they are bisimilar also in c_ρ and d_ρ for every ρ ;
- if c and d use a finite effect algebra of weights, and $\forall \rho. x \sim_k y$ in c_ρ and d_ρ , then $x \sim_k y$ in c and d ;
- x and y are kernel bisimilar in c and d if and only if they are bisimilar in the D_{fc} LTSs where effects are considered as convex functions from density operators to probabilities, i.e. when the adversary can choose a different density operator at each step of the bisimilarity.

In the rest of the section we describe some operators on $\mathbb{E}$ LTSs. When dealing with coalgebraic LTSs, it is typical to define operators like nondeterministic sum, restriction or parallel composition as acting on *processes*, i.e. states of the final coalgebra of the functor under consideration [25]. We instead define operators directly between coalgebras as in [36], allowing us to compose coalgebras of different functors. We postpone to future work how these "macroscopic" operators relates to the "microscopic" ones in the final coalgebra, following the approach of [20]. We present two operators: the first generalizes parallel composition to all $\mathbb{E}$ LTSs, the other is specific to qLTSs and specifies how to perform partial evaluation of the input quantum state. We first introduce the notion of *extensible* graded monad.

In [27], the author proposes a monad graded on a preordered monoid, intended to model the composition of computational side effects and their scope. We replicate this notion in our PCM setting, employing the fact that each PCM automatically carries a preorder structure.

Definition 13 (Extensible Graded Monad). *An M -graded monad is extensible if for all $m \preceq n \in M$, a natural transformation $\xi_{m \preceq n}: T_m \rightarrow T_n$ called extension exists, with $\xi_{m \preceq m} = Id$ and $\xi_{n \preceq o} \circ \xi_{m \preceq n} = \xi_{m \preceq o}$.*

Extensible monads give us a way to canonically extend a computation to a greater grade, which we will need to define parallel composition. All non graded monad are trivially extensible with the identity transformation. We provide an extension for monads built on effect monoids graded on an effect algebra.

Theorem 11. *Suppose $\{\mathbb{E}_m\}$ is an M -graded effect monoid and M an effect algebra, with $-$ its induced difference operator. There is an extensible graded monad $\{D_{\mathbb{E}_m}\}$ with unit $\eta: Id \rightarrow D_{\mathbb{E}_0}$ and multiplication $\mu_{m,n}: D_{\mathbb{E}_m} D_{\mathbb{E}_n} \rightarrow D_{\mathbb{E}_{m+n}}$ defined as in Theorem 4 and extension $\xi_{m \preceq n}$*

$$\xi_{m \preceq n}(\sum_i e_i \bullet x_i) = \sum_i \nabla_{m,n-m}(e_i, 1_{\mathbb{E}_{n-m}}) \bullet x_i.$$

Indeed, Q_C is extensible, since the partial monoid of quantum systems is an effect algebra.

Lemma 2. $\mathcal{S} = \langle \mathcal{P}(\text{Sys}), \emptyset, \uplus, \text{Sys} \setminus \cdot \rangle$ is an effect algebra, with $\setminus$ the usual set difference.

Intuitively, when dealing with a computation consuming the set of qubits C , we can always extend it to a computation on $C' \supseteq C$ by performing the identity measurement on the extra qubits.

We now focus on the parallel composition of $\mathbb{E}\text{LTS}$ s, in which two subsystems can move on their own or exchange messages. In the quantum case, we will be able to compose qLTSs only when their grades are summable, ensuring that they do not perform measurements on the same quantum systems.

In the process-calculi literature, there are different notions of parallel composition, corresponding to different “synchronization styles”: à la CCS, CSP or ACP. Moreover, different extensions have been considered from the original non-deterministic setting to the probabilistic [22, 2] and quantum [8] one. We introduce a *generic parallel operator*, which is parametric both with respect to the synchronization style and to the commutative graded monad chosen for the weights (Id for LTSs, $D_{[0,1]}$ for pLTSs, Q_C for qLTSs). Our parallel operator is defined in two steps: the first specifies how to compose the “non-deterministic structure” of the systems (the $\mathcal{P}^L$ functor), the second how to combine a couple of distributions into a distribution of couples. For the first step, we focus on a CCS-style interleaving composition, formalized as a binary *synchronization operator* $\cdot \parallel \cdot$ on transition functions.

Definition 14 (CCS-Style Synchronization). *Consider the extensible graded monad $\{T_m\}$. Given two coalgebras $X \xrightarrow{c} \mathcal{P}(T_m X)^L$ and $Y \xrightarrow{d} \mathcal{P}(T_n Y)^L$ we define $c|d : X \times Y \rightarrow \mathcal{P}(T_m X \times T_n Y)^L$, the “CCS-style” synchronization of c and d , as*

$$\frac{s \xrightarrow{\mu} \Delta}{s \parallel t \xrightarrow{\mu} \langle \Delta, \xi_{0 \leq n}(\eta(t)) \rangle} \quad \frac{t \xrightarrow{\mu} \Theta}{s \parallel t \xrightarrow{\mu} \langle \xi_{0 \leq m}(\eta(s)), \Theta \rangle} \quad \frac{s \xrightarrow{\mu} \Delta \quad t \xrightarrow{\mu} \Theta}{s \parallel t \xrightarrow{\tau} \langle \Delta, \Theta \rangle}$$

where $\times$ is the cartesian product on **Set**, $s \parallel t$ is an element of $X \times Y$, Δ (resp. Θ) is an element of $T_m X$ (resp. $T_n Y$), and $s \xrightarrow{\mu} \Delta$ is the usual SOS-style notation for $\Delta \in c(s)(\mu)$.

To model CCS-style synchronization we represent the “idle behaviour” of a state s by taking the monad unit and extending it to the desired grade, i.e. $\xi_{0 \leq m}(\eta(s))$. In pLTSs this coincides with the unit $1 \bullet s$. In qLTSs we get the point distribution “scaled” for the given dimension $\mathbb{I}_d \bullet s$. For pure non-deterministic systems, where T is the identity monad, we get the usual parallel composition of CCS.

For the second step, we define a parallel operator $\cdot \parallel \cdot$ on $\mathbb{E}\text{LTS}$ s, in the form of a functor $\mathbf{Set}_{\mathcal{P}(T_m)^L} \times \mathbf{Set}_{\mathcal{P}(T_n)^L} \rightarrow \mathbf{Set}_{\mathcal{P}(T_{m+n})^L}$. The definition is parametric with respect to a natural transformation $\alpha : T_m X \times T_n Y \rightarrow T_{m+n}(X \times Y)$ which specifies how to combine two distributions. For classical, probabilistic and quantum systems, we have commutative monads, which have a canonical transformation α .

Theorem 12. *Given a transformation $\alpha : T_m X \times T_n Y \rightarrow T_{m+n}(X \times Y)$, we can construct a functor $\parallel : \mathbf{Set}_{\mathcal{P}(T_m)^L} \times \mathbf{Set}_{\mathcal{P}(T_n)^L} \rightarrow \mathbf{Set}_{\mathcal{P}(T_{m+n})^L}$ defined by*

$$(X, c) \parallel (Y, d) = (X \times Y, \mathcal{P}(\alpha)^L \circ (c|d)) \quad f \parallel g = f \times g$$

We now consider a special operator acting over qLTS, i.e. the *partial evaluation*.

Definition 15 (Partial Evaluation of Quantum Effects). *Consider a quantum effect $L \in \mathbb{Q}_{\mathcal{H}_C}$. For all quantum state $\rho \in DM_{\mathcal{H}_{C'}}$ with $C' \subseteq C$, we define the partial evaluation of L with state ρ as $tr_{C'}(L(\rho \boxtimes \mathbb{I}))$, where $\mathbb{I}$ is the identity operator on $\mathcal{H}_{C \setminus C'}$ and $\boxtimes$ is the product of Theorem 3.*

Our previously defined total evaluation $\downarrow_\rho$ is of course a specific case of partial evaluation, when $C' = C$. For all ρ , partial evaluation $tr_{C'}(\cdot(\rho \boxtimes \mathbb{I}))$ is an effect morphism, and thus yields a functor from $\mathbf{Set}_{Q_C}$ to $\mathbf{Set}_{Q_{C \setminus C'}}$, and can be extended also to qLTSs via whiskering.

Definition 16 (Partial Evaluation of qLTS). *Let C, C' be collections of quantum systems such that $C' \subseteq C$. For each $\rho \in DM_{\mathcal{H}_{C'}}$ we define the partial evaluation of $\mathbf{Set}^{\mathcal{P}(Q_C)^L}$ with input ρ as the functor $\downarrow_\rho: \mathbf{Set}^{\mathcal{P}(Q_C)^L} \rightarrow \mathbf{Set}^{\mathcal{P}(Q_{C'})^L}$ induced by the effect morphism $L \mapsto tr_{C'}(L(\rho \boxtimes \mathbb{I}))$.*

Thanks to functoriality, parallel composition and partial evaluation preserve bisimilarity.

Theorem 13. *If $s \sim_k t$ then, for all ρ , $s \downarrow_\rho \sim t \downarrow_\rho$. Moreover, if $s' \sim t'$ then $s \parallel s' \sim t \parallel t'$.*

6 Conclusions

We have characterized distributions with weights from a generic effect algebra, subsuming probability and quantum effect distributions. We have introduced monads graded on a partial commutative monoid (PCM) that allow us to grade quantum distributions over their resources, which must be treated linearly, as prescribed by the no-cloning theorem. We have studied effect weighted Markov Chains and labelled transition systems ($\mathbb{E}$ LTS) in a coalgebraic framework, extending previous results about kernel and Aczel-Mendler bisimilarities. We have applied our findings to the quantum setting, proving that each quantum state ρ defines a functor from qLTSs to pLTSs that “instantiates” a quantum process to the probabilistic behaviour it exhibits when the quantum state is ρ . We have compared the two notions of bisimilarity in hand with the desired properties of a behavioural equivalence for quantum systems, proving that the kernel bisimilarity is the only one of the two that captures their observable features, i.e. the induced probabilistic behaviour. Finally, we have defined parallel composition of $\mathbb{E}$ LTSs in a functorial, compositional way, and a partial evaluation operator that given a qLTS instantiates some of its input qubits, paving the way for an $\mathbb{E}$ LTS semantics of quantum process calculi.

Future work We will study the final coalgebra of our functors, investigating the definition of graded GSOS operators for our graded $\mathbb{E}$ LTSs, and their relation with the functorial ones defined in this work. Our framework allows us to compose effects in parallel via the tensor product. We will consider also sequential composition, extending our graded approach to the case of superoperators, which model more general quantum operations. We expect such an extension to come naturally and to preserve our constructions and results. Finally, we will explore the reflection of the kernel bisimilarity in broader cases, i.e. when the quantum effect algebra is finitely generated, but possibly infinite, and when it is numerable in general. Recent work [4] defines bisimulation for Mealy machines with generic effects. We intend to look at the relationship between their effectful bisimilarity and our notion of kernel bisimilarity.

References

- [1] Samson Abramsky, Rui Soares Barbosa, Nadish de Silva & Octavio Zapata (2017): *The Quantum Monad on Relational Structures*, p. 19 pages. doi:10.4230/LIPICS.MFCS.2017.35. arXiv:1705.07310.
- [2] Falk Bartels, Ana Sokolova & Erik de Vink (2004): *A Hierarchy of Probabilistic System Types*. *Theoretical Computer Science* 327(1-2), pp. 3–22, doi:10.1016/j.tcs.2004.07.019.
- [3] Charles H. Bennett & Gilles Brassard (2014): *Quantum Cryptography: Public Key Distribution and Coin Tossing*. *Theoretical Computer Science* 560, pp. 7–11, doi:10.1016/j.tcs.2014.05.025.
- [4] Filippo Bonchi, Elena Di Lavore & Mario Román (2024): *Effectful Mealy machines: bisimulation and trace*. arXiv:2410.10627.
- [5] Filippo Bonchi, Barbara König & Sebastian Küpper (2017): *Up-To Techniques for Weighted Systems*. In Axel Legay & Tiziana Margaria, editors: *Tools and Algorithms for the Construction and Analysis of Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 535–552, doi:10.1007/978-3-662-54577-5_31.

- [6] Filippo Bonchi, Alexandra Silva & Ana Sokolova (2017): *The Power of Convex Algebras*. In: *28th International Conference on Concurrency Theory (CONCUR 2017)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, doi:10.4230/LIPICS.CONCUR.2017.23.
- [7] Marcello Caleffi, Angela Sara Cacciapuoti & Giuseppe Bianchi (2018): *Quantum Internet: From Communication to Distributed Computing!* In: *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication*, ACM, Reykjavik Iceland, pp. 1–4, doi:10.1145/3233188.3233224.
- [8] Lorenzo Ceragioli, Fabio Gadducci, Giuseppe Lomurno & Gabriele Tedeschi (2024): *Effect Semantics for Quantum Process Calculi*. In Rupak Majumdar & Alexandra Silva, editors: *35th International Conference on Concurrency Theory, CONCUR 2024, September 9-13, 2024, Calgary, Canada, LIPIcs 311*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 16:1–16:22, doi:10.4230/LIPICS.CONCUR.2024.16.
- [9] Lorenzo Ceragioli, Fabio Gadducci, Giuseppe Lomurno & Gabriele Tedeschi (2024): *Quantum Bisimilarity Is a Congruence Under Physically Admissible Schedulers*. In Oleg Kiselyov, editor: *Programming Languages and Systems - 22nd Asian Symposium, APLAS 2024, Kyoto, Japan, October 22-24, 2024, Proceedings, Lecture Notes in Computer Science 15194*, Springer, pp. 176–195, doi:10.1007/978-981-97-8943-6_9.
- [10] Lorenzo Ceragioli, Fabio Gadducci, Giuseppe Lomurno & Gabriele Tedeschi (2024): *Quantum Bisimilarity via Barbs and Contexts: Curbing the Power of Non-deterministic Observers*. *Proc. ACM Program. Lang.* 8(POPL), pp. 43:1269–43:1297, doi:10.1145/3632885.
- [11] Bob Coecke & Aleks Kissinger (2017): *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*, 1 edition. Cambridge University Press, doi:10.1017/9781316219317.
- [12] Timothy AS Davidson (2012): *Formal Verification Techniques Using Quantum Process Calculus*. Ph.D. thesis, University of Warwick.
- [13] Yuxin Deng (2018): *Bisimulations for Probabilistic and Quantum Processes (Invited Paper)*. In Sven Schewe & Lijun Zhang, editors: *29th International Conference on Concurrency Theory (CONCUR 2018)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 118, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 2:1–2:14, doi:10.4230/LIPICS.CONCUR.2018.2.
- [14] Yuxin Deng & Yuan Feng (2012): *Open Bisimulation for Quantum Processes*. In Jos C. M. Baeten, Tom Ball & Frank S. de Boer, editors: *Theoretical Computer Science, Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 119–133, doi:10.1007/978-3-642-33475-7_9.
- [15] Yuan Feng, Yuxin Deng & Mingsheng Ying (2014): *Symbolic Bisimulation for Quantum Processes*. *ACM Trans. Comput. Logic* 15(2), pp. 14:1–14:32, doi:10.1145/2579818.
- [16] Yuan Feng & Mingsheng Ying (2015): *Toward Automatic Verification of Quantum Cryptographic Protocols*. In Luca Aceto & David de Frutos Escrig, editors: *26th International Conference on Concurrency Theory (CONCUR 2015)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 42, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, pp. 441–455, doi:10.4230/LIPICS.CONCUR.2015.441.
- [17] Fei Gao, SuJuan Qin, Wei Huang & QiaoYan Wen (2019): *Quantum Private Query: A New Kind of Practical Quantum Cryptographic Protocol*. *Sci. China Phys. Mech. Astron.* 62(7), p. 70301, doi:10.1007/s11433-018-9324-6.
- [18] Stanley Gudder (2008): *Quantum Markov Chains*. *Journal of Mathematical Physics* 49(7), p. 072105, doi:10.1063/1.2953952.
- [19] H. Peter Gumm & Tobias Schröder (2001): *Monoid-Labeled Transition Systems*. *Electronic Notes in Theoretical Computer Science* 44(1), pp. 185–204, doi:10.1016/S1571-0661(04)80908-3.
- [20] Ichiro Hasuo (2011): *The Microcosm Principle and Compositionality of GSOS-Based Component Calculi*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *Algebra and Coalgebra in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 222–236, doi:10.1007/978-3-642-22944-2_16.
- [21] Teiko Heinosaari & Mário Ziman (2011): *The Mathematical Language of Quantum Theory: From Uncertainty to Entanglement*. Cambridge University Press, doi:10.1017/CB09781139031103.
- [22] Matthew Hennessy (2012): *Exploring Probabilistic Bisimulations, Part I*. *Form. Asp. Comput.* 24(4-6), pp. 749–768, doi:10.1007/s00165-012-0242-7.

- [23] Bart Jacobs (2010): *Convexity, Duality and Effects*. In: *Theoretical Computer Science: 6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings 6*, Springer, pp. 1–19, doi:10.1007/978-3-642-15240-5_1.
- [24] Bart Jacobs (2011): *Probabilities, Distribution Monads, and Convex Categories*. *Theoretical Computer Science* 412(28), pp. 3323–3336, doi:10.1016/j.tcs.2011.04.005.
- [25] Bart Jacobs (2016): *Introduction to Coalgebra: Towards Mathematics of States and Observation*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, doi:10.1017/CB09781316823187.
- [26] Jules Jacobs & Thorsten Wißmann (2023): *Fast Coalgebraic Bisimilarity Minimization*. *Proc. ACM Program. Lang.* 7(POPL), pp. 52:1514–52:1541, doi:10.1145/3571245.
- [27] Shin-ya Katsumata (2014): *Parametric effect monads and semantics of effect systems*. In: *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14*, Association for Computing Machinery, New York, NY, USA, p. 633–645, doi:10.1145/2535838.2535846.
- [28] Takahiro Kubota, Yoshihiko Kakutani, Go Kato, Yasuhito Kawano & Hideki Sakurada (2012): *Application of a Process Calculus to Security Proofs of Quantum Protocols*. In: *Proceedings of the International Conference on Foundations of Computer Science (FCS)*, The Steering Committee of The World Congress in Computer Science, Computer ..., p. 1.
- [29] Marie Lalire (2006): *Relations among Quantum Processes: Bisimilarity and Congruence*. arXiv:quant-ph/0603274.
- [30] Gui-lu Long, Fu-guo Deng, Chuan Wang, Xi-han Li, Kai Wen & Wan-ying Wang (2007): *Quantum Secure Direct Communication and Deterministic Secure Quantum Communication*. *Front. Phys. China* 2(3), pp. 251–272, doi:10.1007/s11467-007-0050-3.
- [31] Marino Miculan & Marco Peressotti (2013): *Weak bisimulations for labelled transition systems weighted over semirings*. arXiv:1310.4106.
- [32] Lawrence S. Moss (1999): *Coalgebraic Logic*. *Annals of Pure and Applied Logic* 96(1-3), pp. 277–317, doi:10.1016/S0168-0072(98)00042-6.
- [33] Michael A. Nielsen & Isaac L. Chuang (2010): *Quantum Computation and Quantum Information*, 10th anniversary ed edition. Cambridge University Press, Cambridge ; New York, doi:10.1017/CB09780511976667.
- [34] Ali Ibnun Nurhadi & Nana Rachmana Syambas (2018): *Quantum Key Distribution (QKD) Protocols: A Survey*. In: *2018 4th International Conference on Wireless and Telematics (ICWT)*, pp. 1–5, doi:10.1109/ICWT.2018.8527822.
- [35] Hiroshi Ogawa (2014): *Coalgebraic Approach to Equivalences of Quantum Systems*. Master's thesis, University of Tokyo.
- [36] J.J.M.M. Rutten (2000): *Universal Coalgebra: A Theory of Systems*. *Theoretical Computer Science* 249(1), pp. 3–80, doi:10.1016/S0304-3975(00)00056-6.
- [37] Ana Sokolova & Erik P. de Vink (2004): *Probabilistic Automata: System Types, Parallel Composition and Comparison*, pp. 1–43. Springer Berlin Heidelberg, Berlin, Heidelberg, doi:10.1007/978-3-540-24611-4_1.
- [38] Peiying Zhang, Ning Chen, Shigen Shen, Shui Yu, Sheng Wu & Neeraj Kumar (2022): *Future Quantum Communications and Networking: A Review and Vision*. *IEEE Wireless Commun.*, pp. 1–8, doi:10.1109/MWC.012.2200295.

A Proof Sketches

Theorem 2. *Each effect morphism $m : \mathbb{E} \rightarrow \mathbb{F}$ yields a natural transformation $m \circ \cdot : D_{\mathbb{E}} \Rightarrow D_{\mathbb{F}}$. Moreover, if m is injective then the components of $m \circ \cdot$ are injective.*

Proof sketch. Naturality of $m \circ \cdot$ follows from the fact that m preserves the (partial) sum and that distributions have finite support. If m is injective, the components of $m \circ \cdot$ are also injective because if two different $\Delta, \Theta \in D_{\mathbb{E}}(X)$ differ on $x \in X$, then $m \circ \Delta$ and $m \circ \Theta$ will differ on the same x . $\square$

The evolution of density operators is given as a *trace preserving superoperator* $\mathcal{E} : DM_{\mathcal{H}_A} \rightarrow DM_{\mathcal{H}_B}$, a function defined by its *Kraus operator sum decomposition* $\{E_i\}_i$ for a finite set of indexes $i = 1, \dots, n \times m$, satisfying that $E_i \in \mathbb{C}^{m \times n}$, $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$ and $\sum_i E_i^\dagger E_i = \mathbb{I}_n$, where n and m are the dimension of Hilbert space $\mathcal{H}_A$ and $\mathcal{H}_B$ respectively. The tensor product of density operators $\rho \otimes \sigma$ is defined as their Kronecker product, and of superoperators $\mathcal{E} \otimes \mathcal{F}$ as the superoperator having Kraus decomposition $\{E_i \otimes F_j\}_{i,j}$ with $\{E_i\}_i$ and $\{F_j\}_j$ Kraus decompositions of $\mathcal{E}$ and $\mathcal{F}$.

A morphism between effects is the dual of a superoperator. Let $\mathcal{E}(\rho) = \sum_i E_i \rho E_i^\dagger$ be a superoperator. Its dual is the superoperator $\mathcal{E}^\dagger(L) = \sum_i E_i^\dagger L E_i$.

Theorem 3 (Effect Monoid of Quantum Effect). *Quantum effects carry a commutative $\mathcal{S}$ -graded effect monoid structure, given by:*

- The effect algebra $\mathbb{E}_C$ is $\mathbb{Q}_{\mathcal{H}_C}$ for all collection of systems $C \in \mathcal{P}(\text{Sys})$
- The operator $\nabla_{C,D} : \mathbb{E}_C \otimes \mathbb{E}_D \rightarrow \mathbb{E}_{C \uplus D}$ is denoted as $\boxtimes$ and defined by $L_1 \boxtimes L_2 = \text{Sort}_{C,D}(L_1 \otimes_k L_2)$, where $\otimes_k$ is the Kronecker product between $L_1 \in \mathbb{Q}_{\mathcal{H}_C}$ and $L_2 \in \mathbb{Q}_{\mathcal{H}_D}$, and $\text{Sort}_{C,D}$ is the unitary transformation which “sorts” the Hilbert Space $\mathcal{H}_C \otimes \mathcal{H}_D$ into $\mathcal{H}_{C \uplus D}$.
- The effect morphism $\eta : 2 \rightarrow \mathbb{E}_\emptyset$ defined by $\{0 \mapsto 0, 1 \mapsto 1\}$.

Proof sketch. Recall that $\text{Sort}_{C,D}$ is the transformation on quantum effects (i.e. a dual superoperator) which applies a unitary permutation matrix from $\mathcal{H}_C \otimes \mathcal{H}_D$ to $\mathcal{H}_{C \uplus D}$. Unitality follows from the definition of Kronecker product and from $\text{Sort}_{C,\emptyset}$ being the identity transformation. For commutativity, we check that $\text{Swap}_{C,D}$, the superoperator that permutes $\mathcal{H}_C \otimes \mathcal{H}_D$ in $\mathcal{H}_D \otimes \mathcal{H}_C$, respects sorting:

$$\text{Sort}_{D,C}(L_2 \otimes L_1) = \text{Sort}_{D,C}(\text{Swap}_{C,D}(L_1 \otimes L_2)) = (\text{Sort}_{D,C} \circ \text{Swap}_{C,D})(L_1 \otimes L_2) = \text{Sort}_{C,D}(L_1 \otimes L_2).$$

For associativity, we have that

$$\text{Sort}_{C,D \uplus E}(L_1 \otimes \text{Sort}_{D,E}(L_2 \otimes L_3)) = (\text{Sort}_{C,D \uplus E} \circ (\text{Id} \otimes \text{Sort}_{D,E}))(L_1 \otimes L_2 \otimes L_3) = \text{Sort}_{C,D,E}(L_1 \otimes L_2 \otimes L_3)$$

and similarly for $\text{Sort}_{C \uplus D,E}(\text{Sort}_{C,D}(L_1 \otimes L_2) \otimes L_3)$, where Id is the identity superoperator, and $\text{Sort}_{C,D,E}$ is the superoperator going from $\mathcal{H}_C \otimes \mathcal{H}_D \otimes \mathcal{H}_E$ to $\mathcal{H}_{C \uplus D \uplus E}$. $\square$

Theorem 4 (Graded Monads of Graded Effect Monoids). *If $\{\mathbb{E}_m\}$ is an M -graded effect monoid, there is a graded monad $\{D_{\mathbb{E}_m}\}$ with unit $\eta : \text{Id} \rightarrow D_{\mathbb{E}_0}$ and multiplication $\mu_{m,n} : D_{\mathbb{E}_m} D_{\mathbb{E}_n} \rightarrow D_{\mathbb{E}_{m+n}}$ given by*

$$\eta(x) = 1_{\mathbb{E}_0} \bullet x \quad \mu_{m,n}(\sum_i e_i \bullet \Delta_i)x = \sum_i \nabla_{m,n}(e_i, \Delta_i(x))$$

Proof sketch. The proof proceeds as in the total case: the unit and multiplication of the monad are defined with the graded monoid structure of the weights, and the associativity and unitality conditions follow from the graded monoid ones. $\square$

Theorem 5. *If $\{\mathbb{E}_m\}$ is a commutative graded effect monoid, then $\{D_{\mathbb{E}_m}\}$ is a commutative graded monad with its canonical strength.*

Proof sketch. As any **Set**-endofunctor, $D_{\mathbb{E}_m}$ has a canonical left-strength $\sigma_{m,X,Y} : X \times D_{\mathbb{E}_m} Y \rightarrow D_{\mathbb{E}_m}(X \times Y)$ given by

$$(x, (\sum_i e_i \bullet y_i)) \mapsto \sum_i e_i \bullet x, y_i$$

and a corresponding right strength given by the monoidal structure of **Set**. Both transformations yield indeed a strong graded monad. It's easy to see that the morphism $\mu_{m,n} \circ D_{\mathbb{E}_m} \sigma_n \circ \tau_m$ brings $(\sum_i e_i \bullet x_i, \sum_j e'_j \bullet y_j)$ into $\sum_{i,j} \nabla_{n,m}(e'_j, e_i) \bullet (x, y)$, while $\mu_{n,m} \circ D_{\mathbb{E}_n} \tau_m \circ \sigma_n$ brings it to $\sum_{i,j} \nabla_{m,n}(e_i, e'_j) \bullet (x, y)$. The two coincide whenever ∇ is commutative. $\square$

Theorem 6. *Let $\mathbb{E}$ be an effect algebra. Let $X \xrightarrow{c} D_{\mathbb{E}} X$ and $Y \xrightarrow{d} D_{\mathbb{E}} Y$ be two $D_{\mathbb{E}}$ -coalgebras. For all $x \in X$ and $y \in Y$: (i) $x \sim_{AM} y \implies x \sim_k y$, and (ii) $x \sim_k y \implies x \sim_{AM} y$ if and only if $\mathbb{E}$ is decomposable.*

Proof sketch. The first point has been demonstrated in [2, Corollary 8].

The *if* case of the second point follows from the composition of two previously demonstrated results. In [36, Theorem 5.13], the authors prove that if a commutative monoid is positive and decomposable then the functor $D_{\mathbb{E}}$ weakly preserves weak pullbacks. Such demonstration is also applicable to partial commutative monoid, of which effect algebras are a special case. Again, [2, Corollary 8] prove that if $D_{\mathbb{E}}$ preserves weak pullback then $\sim_k$ implies $\sim_{AM}$.

Finally, the *only if* case is demonstrated through contraposition. Assume we have effects $a, b, c, d \in \mathbb{E}$ that are not decomposable, i.e. $a + b = c + d$ and there are no $e_{11}, e_{12}, e_{21}, e_{22} \in \mathbb{E}$ such that $a = e_{11} + e_{12}$, $b = e_{21} + e_{22}$, $c = e_{11} + e_{21}$ and $d = e_{12} + e_{22}$. Let $s = a + b = c + d$.

Consider three coalgebras:

1. (X, c) with $X = \{x_1, x_2, x_3\}$ and $c(x_1) = \{x_2 \mapsto a, x_3 \mapsto b\}$.
2. (Y, d) with $Y = \{y_1, y_2, y_3\}$ and $d(y_1) = \{y_2 \mapsto c, y_3 \mapsto d\}$.
3. (Z, z) with $Z = \{z_1, z_2\}$ and $z(z_1) = \{z_2 \mapsto s\}$.

It is straightforward to show that $X \xrightarrow{m_1} Z \xleftarrow{m_2} X$ with $m_1(x_1) = m_2(y_1) = z_1$ and $m_1(x_2) = m_1(x_3) = m_2(y_1) = m_2(y_2) = z_2$, $m_1(x_4) = m_2(x_4) = z_3$ is a cocongruence, therefore $x_1 \sim_k y_1$. By [36, Lemma 5.5], a relation $R \subseteq X \times Y$ is an AM-bisimulation if and only if for every $(a, b) \in R$ there is a $|X| \times |Y|$ matrix, $(m_{x,y})$, with entries from $\mathbb{E}$ such that

- there are all but finitely many $m_{x,y} = 0$;
- if $m_{x,y} \neq 0$ then $(x, y) \in R$;
- $\forall x \in X. c(a)(x) = \sum_{y \in Y} m_{x,y}$;
- $\forall y \in Y. d(b)(y) = \sum_{x \in X} m_{x,y}$;

Let us consider the case for the pair (x_1, y_1) . The requirements can be represented by the following table.

0	0	0	0
0	m_{x_2, y_2}	m_{x_2, y_3}	a
0	m_{x_3, y_2}	m_{x_3, y_3}	b
0	c	d	

However, by assumption of non-decomposability there are no $m_{x_2, y_2}, m_{x_2, y_3}, m_{x_3, y_2}, m_{x_3, y_3}$ that can satisfy such requirements, hence $x_1 \not\sim_{AM} y_1$. $\square$

Theorem 8. *Let $\alpha : F \Rightarrow G$ be a natural transformation between two functors F and $G : \mathbf{Set} \rightarrow \mathbf{Set}$. If all components α_x of α are injective then the induced functor $\alpha \circ \cdot$ reflects the kernel bisimilarity.*

Proof sketch. Let $X \xrightarrow{m_1} Z \xleftarrow{m_2} Y$ be a cocongruence between any two coalgebras $X \xrightarrow{\alpha_X \circ c} GX$ and $Y \xrightarrow{\alpha_Y \circ d} GY$. Let R be the pullback in $\mathbf{Set}$ of such cospan, witnessing the bisimilarity between the two G -coalgebras. [2, Theorem 5] prove that the same cocongruence is also witness for the kernel bisimulation between the coalgebras $X \xrightarrow{c} FX$ and $Y \xrightarrow{d} FY$. Therefore, R is also witness of the bisimilarity between the two F -coalgebras. $\square$

Proposition 1. *If the dimension of $\mathcal{H}$ is greater or equal than 2, then $\mathbb{Q}_{\mathcal{H}}$ is not decomposable.*

Proof sketch. Take the following equality $|0\rangle\langle 0| + |1\rangle\langle 1| = |+\rangle\langle +| + |-\rangle\langle -|$. By [21, Proposition 1.63], if R is a positive rank-1 operator, T a positive operator and $T \subseteq R$, then $T = pR$ for some $p \in [0, 1]$. Recall that the partial order of quantum effects is the Löwner order, and that the considered effects are positive rank-1 operator. Let $|0\rangle\langle 0| = e_{11} + e_{12}$ for some $e_{11}, e_{12} \in \mathcal{H}_C$. It must be that $e_{11} = p_{11}|0\rangle\langle 0|$ for some $p_{11} \in [0, 1]$, and similarly $e_{12} = p_{12}|0\rangle\langle 0|$, with $p_{11} + p_{12} = 1$. But in order to satisfy the decomposability requirement, $|+\rangle\langle +| = e_{11} + e_{21}$ for some $e_{21} \in \mathcal{H}_C$. Then, $e_{11} = p'_{11}|+\rangle\langle +|$ for some $p'_{11} \in [0, 1]$. Thus, $p_{11}|0\rangle\langle 0| = p'_{11}|+\rangle\langle +|$, which holds if and only if $p_{11} = p'_{11} = 0$. Assume without loss of generality that $e_{11} = 0$. The same line of reasoning can show that $p_{12}|0\rangle\langle 0| = p'_{12}|-\rangle\langle -|$ holds if and only if $p_{12} = p'_{12} = 0$. Hence, both e_{11} and e_{12} must be 0, thus it is impossible to decompose $|0\rangle\langle 0| + |1\rangle\langle 1| = |+\rangle\langle +| + |-\rangle\langle -|$. This construction generalizes easily to dimensions greater than two. $\square$

Theorem 9. *Let (X, c) and (Y, d) be two $D_{\mathbb{L}}$ -coalgebras, and let $x \in X$ and $y \in Y$. If x and y are kernel bisimilar in $\downarrow_{\rho}(X, c)$ and $\downarrow_{\rho}(Y, d)$ for all ρ , then x and y are kernel bisimilar in (X, c) and (Y, d) .*

Proof sketch. Assume x and y are kernel bisimilar in $\downarrow_{\rho}(X, c)$ and $\downarrow_{\rho}(Y, d)$ for all ρ . By Lemma 1 a $\hat{\rho}$ such that the effect algebra homomorphism $L \mapsto \text{tr}(L\hat{\rho})$ is injective. Then, the components of the natural transformation induced by this homomorphism are injective Theorem 2. By assumption, x and y are bisimilar also in $\downarrow_{\hat{\rho}}(X, c)$ and $\downarrow_{\hat{\rho}}(Y, d)$. The result then follows by Theorem 7, as $\downarrow_{\hat{\rho}}$ reflects kernel bisimilarity. $\square$

Theorem 10. *Let (X, c) and (Y, d) be Q_C -coalgebras, and let $x \in X$ and $y \in Y$. Then $x \sim_k y$ in (X, c) and (Y, d) holds if and only if $x \sim_k y$ holds in $(\alpha_{lpp} \circ \cdot)(X, c)$ and $(\alpha_{lpp} \circ \cdot)(Y, d)$.*

Proof sketch. Assume x and y are kernel bisimilar in (X, c) and (Y, d) , then the injective effect algebra homomorphism $L \mapsto \lambda \rho. \text{tr}(L\rho)$ defines a natural transformation $\alpha_{lpp} : Q_C \rightarrow D_{f_c}$ with injective components Lemma 1. By Theorem 7, $\alpha_{lpp} \circ \cdot$ is a functor from $\mathbf{Set}_{Q_C}$ to $\mathbf{Set}_{f_c}$: it preserves both bisimilarities as every functor, moreover, it also reflects kernel bisimilarity since $\alpha_{lpp} : Q_C \rightarrow D_{f_c}$ has injective components. $\square$

Theorem 11. *Suppose $\{\mathbb{E}_m\}$ is an M -graded effect monoid and M an effect algebra, with $-$ its induced difference operator. There is an extensible graded monad $\{D_{\mathbb{E}_m}\}$ with unit $\eta : Id \rightarrow D_{\mathbb{E}_0}$ and multiplication $\mu_{m,n} : D_{\mathbb{E}_m} D_{\mathbb{E}_n} \rightarrow D_{\mathbb{E}_{m+n}}$ defined as in Theorem 4 and extension $\xi_{m \leq n}$*

$$\xi_{m \leq n}(\sum_i e_i \bullet x_i) = \sum_i \nabla_{m,n-m}(e_i, 1_{\mathbb{E}_{n-m}}) \bullet x_i.$$

Proof sketch. The desired conditions follow from the unitality and associativity of ∇ , and from the fact that $\nabla_{m,n}(1, 1) = 1$, since it is an effect bihomomorphism. $\square$

Lemma 2. $\mathcal{S} = \langle \mathcal{P}(\text{Sys}), \emptyset, \uplus, \text{Sys} \setminus \cdot \rangle$ is an effect algebra, with $\setminus$ the usual set difference.

Proof sketch. $\text{Sys} \setminus C_1 = C_2$ is the unique element in $\mathcal{P}(\text{Sys})$ such that $C_1 \uplus C_2 = \text{Sys} = \text{Sys} \setminus \emptyset$ by definition. Finally, by construction, $C \uplus \text{Sys}$ is defined only if $C \cap \text{Sys} = \emptyset$, i.e. C must be $\emptyset$. $\square$

Lemma 3. Given two coalgebra homomorphisms $f : c \rightarrow c', g : d \rightarrow d'$, the synchronization operator $\cdot | \cdot$ makes the following diagram commute

$$\begin{array}{ccc} X \times Y & \xrightarrow{f \times g} & X' \times Y' \\ \downarrow c|d & & \downarrow c'|d' \\ \mathcal{P}(T_m X \times T_n Y)^L & \xrightarrow{\mathcal{P}(T_m f \times T_n g)^L} & \mathcal{P}(T_m X' \times T_n Y')^L \end{array}$$

Proof sketch. The commutativity of the above diagram correspond to verifying that

- For all $s \parallel t \xrightarrow{\mu} \langle \Delta, \Theta \rangle$, there exists Δ', Θ' such that $f(s) \parallel g(t) \xrightarrow{\mu} \langle \Delta', \Theta' \rangle$ and $\Delta' = f(\Delta), \Theta' = g(\Theta)$
- For all $f(s) \parallel g(t) \xrightarrow{\mu} \langle \Delta', \Theta' \rangle$, there exists Δ, Θ such that $s \parallel t \xrightarrow{\mu} \langle \Delta, \Theta \rangle$ and $\Delta' = f(\Delta), \Theta' = g(\Theta)$

which we can prove by cases on the definition on $\cdot | \cdot$, thanks to the hypothesis on f , which correspond to

- For all $s \xrightarrow{\mu} \Delta$, there exists Δ' such that $f(s) \xrightarrow{\mu} \Delta'$ and $\Delta' = f(\Delta)$
- For all $f(s) \xrightarrow{\mu} \Delta'$, there exists Δ such that $s \xrightarrow{\mu} \Delta$ and $\Delta' = f(\Delta)$

and similarly for g . $\square$

Theorem 12. Given a transformation $\alpha : T_m X \times T_n Y \rightarrow T_{m+n}(X \times Y)$, we can construct a functor $\parallel : \mathbf{Set}_{\mathcal{P}(T_m)^L} \times \mathbf{Set}_{\mathcal{P}(T_n)^L} \rightarrow \mathbf{Set}_{\mathcal{P}(T_{m+n})^L}$ defined by

$$(X, c) \parallel (Y, d) = (X \times Y, \mathcal{P}(\alpha)^L \circ (c|d)) \quad f \parallel g = f \times g$$

Proof sketch. Checking that $\parallel$ is a functor amounts to checking the commutativity of the following diagram

$$\begin{array}{ccc} X \times Y & \xrightarrow{f \times g} & X' \times Y' \\ \downarrow c|d & & \downarrow c'|d' \\ \mathcal{P}(T_m X \times T_n Y)^L & \xrightarrow{\mathcal{P}(T_m f \times T_n g)^L} & \mathcal{P}(T_m X' \times T_n Y')^L \\ \downarrow \mathcal{P}(\alpha)^L & & \downarrow \mathcal{P}(\alpha)^L \\ \mathcal{P}(T_{m+n}(X \times Y))^L & \xrightarrow{\mathcal{P}(T_{m+n}(f \times g))^L} & \mathcal{P}(T_{m+n}(X' \times Y'))^L \end{array}$$

which holds since the bottom square commutes by naturality of α , and the top squares commute by Lemma 3. $\square$

Theorem 13. If $s \sim_k t$ then, for all ρ , $s \downarrow_\rho \sim t \downarrow_\rho$. Moreover, if $s' \sim t'$ then $s \parallel s' \sim t \parallel t'$.

Proof sketch. For the first, notice that $L \mapsto L \downarrow_\rho$ is an effect morphism thanks to the linearity of the partial trace. Thus, it yields a natural transformation and a functor by Theorem 2 and Theorem 7, and functors preserve bisimilarity. For the second, thanks to functoriality, we know that $\parallel$ maps cospans in cospans:

$$\left(\begin{array}{ccc} (X, c) & & (Y, d) \\ \downarrow f & \swarrow g & \\ (Z, z) & & \end{array} \parallel \begin{array}{ccc} (X', c') & & (Y', d') \\ \downarrow f' & \swarrow g' & \\ (Z', z') & & \end{array} \right) = \begin{array}{ccc} (X, c) \parallel (X', c') & & (Y, d) \parallel (Y', d') \\ \downarrow f \times g & \swarrow f' \times g' & \\ (Z, z) \parallel (Z', z') & & \end{array}$$

Then, it is easy to see that the **Set**-pullback of $X \times X' \rightarrow Z \times Z' \leftarrow Y \times Y'$ contains all and only the couples $s \parallel s', t \parallel t'$ such that $f(s) = g(t)$ and $f'(s') = g'(t')$. In other words, the cartesian product of two bisimulations is a bisimulation. $\square$

Reinforcement Learning in Categorical Cybernetics

Jules Hedges

Riu Rodríguez Sakamoto

We show that several major algorithms of reinforcement learning (RL) fit into the framework of categorical cybernetics, that is to say, parametrised bidirectional processes. We build on our previous work in which we show that value iteration can be represented by precomposition with a certain optic. The outline of the main construction in this paper is: (1) We extend the Bellman operators to parametrised optics that apply to action-value functions and depend on a sample. (2) We apply a representable contravariant functor, obtaining a parametrised function that applies the Bellman iteration. (3) This parametrised function becomes the backward pass of another parametrised optic that represents the model, which interacts with an environment via an agent. Thus, parametrised optics appear in two different ways in our construction, with one becoming part of the other. As we show, many of the major classes of algorithms in RL can be seen as different extremal cases of this general setup: dynamic programming, Monte Carlo methods, temporal difference learning, and deep RL. We see this as strong evidence that this approach is a natural one and believe that it will be a fruitful way to think about RL in the future.

1 Introduction

Reinforcement learning (RL) refers to a class of methods in machine learning for optimising a long-run reward during interaction with an unknown environment. It is considered one of the major pillars of machine learning, along with *deep learning* (neural networks and differentiable programming), *unsupervised learning* (statistical clustering methods, which includes topological data analysis [Ghr08]) and *variational learning* (Bayesian inference and related probabilistic methods). It can be seen as an extension of *dynamic programming* methods in optimal control theory [Ber19], which drops the assumption that a model of the environment is known. RL, combined with deep learning methods to produce *deep RL*, notably achieved state of the art success in practical game playing, with AlphaGo [SSS⁺17] defeating the human Go champion in 2016 and AlphaStar [VBC⁺19] achieving Grandmaster status in the real time strategy game StarCraft II.

In this paper we show that several major algorithms of reinforcement learning fit into the framework of *categorical cybernetics*, that is to say, *parametrised bidirectional processes* [CGHR22]. This branch of applied category theory has already been applied to deep learning [CGG⁺22, Gav24], variational learning [BHS23, Smi23] and game theory [GHWZ18, BHZ23]. It is also a close relative of the *categorical systems theory* of Myers, Spivak and others [Mye23, NS24].

We build on our previous work [HS23] in which we show that *value iteration*, a fundamental method common to both dynamic programming and RL, can be represented (in the technical sense) by precomposition with a certain optic. Specifically, for each *policy* π we define an *optic* $\mathbb{B}(\pi) : (S_{\mathbb{R}}) \rightarrow (S_{\mathbb{R}})$, where S is the set of states of a Markov decision process. This has the property that for any *value function* $V : S \rightarrow \mathbb{R}$, represented as an optic $V : (S_{\mathbb{R}}) \rightarrow I$, $V \circ \mathbb{B}(\pi)$ is a better value function. This precomposition with $\mathbb{B}(\pi)$ is called a *Bellman operator*. This bidirectional approach differs from Bakirtzis, Savvas and Topcu’s categorical specification of MDPs [BST22], which focuses on several compositional aspects

such as subprocesses and sequential tasks. Our subject matter is the structure of the algorithms that are used in such RL environments and how they relate to each other.

The outline of the main construction in this paper is: (1) We extend $\mathbb{B}$ to a *parametrised optic* representing a more general class of Bellman operators that apply to *action-value functions* and depend on a *sample* as a parameter. (2) We apply $\mathbb{K}$, a representable contravariant functor that already plays a foundational role in compositional game theory, obtaining a parametrised function $\mathfrak{B} = \mathbb{K}(\mathbb{B})$ that applies the Bellman iteration. (3) This parametrised function becomes the backward pass of another parametrised optic that represents the *model*, which interacts with an *environment* via an *agent*. Thus, parametrised optics appear in two different ways in our construction, with one becoming part of the other. This stays within the existing ingredients of categorical cybernetics, but combines them in a way that has not been seen elsewhere.

As we show, many of the major classes of algorithms in RL can be seen as extremal cases of this general setup: dynamic programming, Monte Carlo methods, temporal difference learning, and deep RL. We see this as strong evidence that this approach is a natural one and believe that it will be a fruitful way to think about RL in the future. For now our goal is merely to achieve a better conceptual understanding of RL, although the hope is that this will eventually translate into quantifiable benefits such as improved modelling techniques. Although we focus on single-agent RL, the compositionality of our methods makes them naturally well-suited to *multi-agent* RL, which is a close relative of game theory.

2 Background: Reinforcement learning

Algorithms in RL specify how agents learn optimal behaviours through interaction with their environment. This interaction provides feedback to actions, and is the key feature that differentiates it with respect to supervised and unsupervised learning. The fundamental goal of RL is to enable agents to make sequential decisions in dynamic environments to maximize long-term cumulative rewards. This process involves the agent taking actions, observing the resulting states and rewards, and using this information to update its decision-making strategy over time.

Our approach to study these algorithms is structural, and the main structural distinction is between the agent and the environment. The **environment** represents the external system with which the agent interacts, and is assumed to be a Markov decision process. To quickly recall, a Markov process (MP) consists of a set of states S and a stochastic transition function $t : S \rightarrow DS$, where D is some probability monad over **Set**. A Markov reward process (MRP) is a Markov process with an additional function $r : S \rightarrow D\mathbb{R}$ that outputs the immediate *reward* for the current state. This reward function can be in general be correlated with the transition, in which case we write $t : S \rightarrow D(S \times \mathbb{R})$. When clear from context, we will abuse the notation and write r for the reward of a particular state, e.g. $(s', r) \sim t(s)$. A Markov decision process (MDP) is a MRP with a set A of actions, whose transition and reward functions now depend on the action taken at each state too, and is also in general correlated too $t : S \times A \rightarrow D(S \times \mathbb{R})$.¹ An agent's goal is to maximize the *expected long-run reward* $\sum \gamma^i r(s_i)$, where $0 < \gamma \leq 1$ is a hyperparameter called the **discount factor** which controls the agent's "patience", or preference between rewards in the present and rewards in the future.

The environment's response to an agent's action is given by transition dynamics that can be assumed to have the Markov property, and the environment's state is known to the agent. When the agent only has access to a partial observation of the state, we speak of a partially observable MDP (POMDP).

¹The cases where the reward function is decorrelated with t as in $S \times A \rightarrow D(\mathbb{R})$, $S \rightarrow D(\mathbb{R})$, or $S \times A \times S \rightarrow D(\mathbb{R})$ can be embedded in our modelling choice for t .

The **agent** has as core components the policy, the reward, the value function and the internal model. A **policy** or scheduler $\pi : S \rightarrow TA$ defines its strategy mapping states to actions. The policy is either single-valued or deterministic ($T = 1$ the identity functor), many-valued ($T = \mathcal{P}$ the powerset functor, like argmax) or probabilistic ($T = D$ the distribution functor, like ε -greedy), with probabilistic being the most common. The **reward** is the immediate response of the environment after an action, and the maximization of its expected cumulative sum is the goal of the agent under the *reward hypothesis* [SSPS21]. A **value function** estimates this expected long-term reward associated with following a particular policy. Usually one works with either a state value function $V : S \rightarrow \mathbb{R}$ or a state-action value function $Q : S \times A \rightarrow \mathbb{R}$, where $V(s)$ estimates the long-run reward of following a certain policy from each state, and $Q(s, a)$ estimates the long-run reward of taking each action in each state and then following a certain policy after that. Both policies and value functions are characterised as solutions of functional equations known as **Bellman equations**, using the temporal ‘self-similarity’ of MDPs.

When S and A are finite sets the function Q is typically implemented as a mutable lookup table called a Q-table or Q-matrix. A **model** is an approximation or representation of the environment’s dynamics, allowing the agent to simulate or predict future states and rewards. One surrogate objective of an agent is to improve its model. Not all agents have models, so there’s a distinction between **model-based** and **model-free** methods.

Methods whose policies for environment interaction π_{beh} (“behaviour policy”) are different to the ones for model improvement π_{tgt} (“target policy”) are called **off-policy**. **On-policy** methods only have a single policy. Finally, another distinction is drawn between **online** and **offline** or **batch RL** methods, where the former family learns while interacting with the environment, while the latter learns from pre-recorded experiences.

RL encompasses many algorithms and methodologies, including dynamic programming, Monte Carlo methods, temporal difference learning, deep reinforcement learning, and more. This diversity of methods employs experimental and formal justifications to tackle weak spots in this learning theory such as the credit-assignment problem, the exploration-exploitation tradeoff and coping with state that is hidden or too big to represent explicitly. Many of these were problems already identified in preceding fields such as psychology and neuroscience [KLM96].

2.1 Dynamic programming

Dynamic programming (DP) methods are an idealized class of model-based algorithms that do not need to interact with the environment because they have a perfect model of it as an MDP. They are not usually used in their classical formulation that we describe next in practical settings because of the perfect model assumption and their high computational expense, and serve rather as a theoretical baseline to approximation methods and other RL techniques.

The idea behind DP is to treat the Bellman equation for the optimal value of a policy and the Bellman equation for the optimal policy of a value function as update operators on a space of value functions.

The search for an optimal policy happens entirely within the agent’s model, interleaving two feedback operations called the *value improvement* and *policy improvement* steps which treat the Bellman equations as update rules. This process of updating previous estimates is called **bootstrapping**.

- **Value improvement** or policy evaluation updates the value function $V : S \rightarrow \mathbb{R}$ pointwise by traversing the state space S and updating the state’s estimated value $V(s)$ with the expected dis-

counted value after one simulation step:

$$V(s) \leftarrow \mathbb{E}_{\substack{a \sim \pi(s) \\ (s', r) \sim t(s, a)}} [r + \gamma V(s')] = \sum_{a \in A} \pi(a | s) \sum_{\substack{s' \in S \\ r \in \mathbb{R}}} t(s', r | s, a) \cdot (r + \gamma V(s')) \quad (1)$$

Here we are interchangeably considering a stochastic function $f : X \rightarrow DY$ as a function $f : Y \times X \rightarrow [0, 1]$, whose causality is still reflected by the traditional middle bar $f(y | x)$. The sum over $\mathbb{R}$ makes sense when D is finite support probability distributions, and in more general settings is replaced with an integral. We write $\mathfrak{B}_{\text{val}}(V, \pi)(s) = \mathbb{E}[r + \gamma V(s')]$ for the operator $\mathfrak{B}_{\text{val}} : \mathbb{R}^S \times (TA)^S \rightarrow \mathbb{R}^S$. We will discuss Bellman operators in Section 5.

- **Policy improvement** updates the policy function $\pi : S \rightarrow TA$ pointwise by traversing the state space S and updating the action taken in the state $\pi(s)$ with $\arg\max_a \mathbb{E}_{(s', r) \sim t(s, a)} [r + \gamma V(s')]$. Similarly, we write $\mathfrak{B}_{\text{pol}}(V)(s) = \arg\max_a \mathbb{E}_{s', r \sim t(s, a)} [r + \gamma V(s')]$ for the operator $\mathfrak{B}_{\text{pol}} : \mathbb{R}^S \rightarrow (TA)^S$.

Depending on the sequencing of these two steps, we have three classic algorithms, where we write $(-)^{\dagger}$ for the (in practice approximate) fixpoint of the operator and, respectively, $\overline{\mathfrak{B}_{\text{pol}}}(V, \pi) = (V, \mathfrak{B}_{\text{pol}}(V))$ and $\overline{\mathfrak{B}_{\text{val}}}(V, \pi) = (\mathfrak{B}_{\text{val}}(V, \pi), \pi)$ for the embeddings of the two Bellman operators as maps $\mathbb{R}^S \times (TA)^S \rightarrow \mathbb{R}^S \times (TA)^S$: **policy iteration** (PIT) as $(\overline{\mathfrak{B}_{\text{pol}}} \circ \overline{\mathfrak{B}_{\text{val}}})^{\dagger}$, **value iteration** (VIT) as $(\overline{\mathfrak{B}_{\text{pol}}} \circ \overline{\mathfrak{B}_{\text{val}}})^{\dagger}$ and **generalized policy iteration** (GPI) as $(\overline{\mathfrak{B}_{\text{pol}}}^m \circ \overline{\mathfrak{B}_{\text{val}}}^n)^{\dagger}$ for $m, n > 0$.

2.2 Monte Carlo

Monte Carlo (MC) methods are antithetical to DP, because they don't assume any prior knowledge of the environment's dynamics. Without this knowledge, the way to learn the value function and obtain a optimal policy is to estimate it from sample trajectories. Averaging over many trajectories should converge to the expected value.

The agent's internal model consists of a value function $Q : S \times A \rightarrow \mathbb{R}$, from which a policy like the ε -greedy $\pi : S \rightarrow DA$ is derived: $\pi(s) = \arg\max_a Q(s, a)$ with probability $1 - \varepsilon$ and uniformly random between all actions with probability ε . The value function improvement is pointwise, but unlike DP, MC improves $Q(s, a)$ by averaging over many returns that start at (s, a) . Given a single episode $(s, a, r, s', a', r', \dots)$ starting at (s, a) , the update **target** becomes $G = \sum_t \gamma^t r_t$, and the value function updates as

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha G \quad (2)$$

where the learning rate $\alpha : [0, 1]$ is a step size hyperparameter. Note that the lack of bootstrapping is shown by the fact that G does not contain any reference to the value function.

2.3 Temporal difference learning

Temporal difference learning (TD) is a class of methods that learn from both the interaction with the environment (MC's sampling) and from previous estimates of the value function (DP's bootstrapping).

Given a finite episode $(s, a, r, \dots, s_n, a_n)$ starting at (s, a) , we can modify the target for (2) to consist of the discounted sum of the $n - 1$ returns and an estimated long-run value of the last state-action pair. We write n -TD for the class of TD methods whose trajectories contain n return values.

Example 2.1 (SARSA [Sut95]). SARSA is a 1-TD on-policy control method, which updates the (s, a) -indexed Q-value with the target $G = r + \gamma Q(s', a')$. The name originates from the model feedback consisting of a 1-step episode (s, a, r, s', a') . Some variants of SARSA include n -SARSA, with $G = \sum_{t=1}^{n-1} \gamma^t r_t + \gamma^n Q(s_n, a_n)$, and Exp-SARSA, with $G = r + \gamma \mathbb{E}_{a \sim \pi_{\text{tgt}}(s)} Q(s, a)$, which is off-policy because the last action is determined in expectation by a target policy π_{tgt} .

Example 2.2 (Q-learning [WD92]). In Q-learning, given the current state s the agent performs an action $a \sim \pi_{\text{beh}}(s)$ using a policy derived from its internal Q-table, for example an ε -greedy policy, and gets from the environment the reward r and the next state s' . The feedback to the model is the tuple (s, a, r, s') . The model then updates its Q-table with its target policy $G = r + \gamma Q(s, \pi_{\text{tgt}}(s)) = r + \gamma \max_{a' \in A} Q(s', a')$. It is an off-policy method because the last action used to compute the update is $\pi_{\text{tgt}}(s') = \arg\max_{a' \in A} Q(s', a')$ and not $\pi_{\text{beh}}(s')$.

Q-learning is the first appearance of a major subtlety of RL: the distinction between actions that the agent actually performs during an interaction with its environment, and actions which are “internal” or “simulated”. The actions that the agent actually performs in Q-learning are always drawn from the policy π_{beh} , whereas the action $\arg\max_{a' \in A} Q(s', a')$ is used only when computing updates. We can consider this to be a separate target policy, $\pi_{\text{tgt}}(s') = \arg\max_{a' \in A} Q(s', a')$.

3 Background: categorical cybernetics

In this section we quickly recall the main ideas of categorical cybernetics, mostly from [CGHR22].

3.1 Actegories

Given a monoidal category $\mathcal{M}$ and a category $\mathcal{C}$, an action of $\mathcal{M}$ on $\mathcal{C}$, also called an **actegory**, is a functor $\bullet : \mathcal{M} \times \mathcal{C} \rightarrow \mathcal{C}$ together with coherent isomorphisms $I \bullet X \cong X$ and $(M \otimes N) \bullet X \cong M \bullet (N \bullet X)$ [CG23]. Every monoidal category has a self-action given by $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$.

If $\mathcal{M}$ and $\mathcal{C}$ are monoidal categories and $F : \mathcal{M} \rightarrow \mathcal{C}$ is a strong monoidal functor, then $M \bullet X = F(M) \otimes X$ is an actegory. A coherent action of one symmetric monoidal category on another, called a **symmetric actegory**, is necessarily of this form. For example, the self-action of a symmetric monoidal category is a symmetric actegory given by the identity functor $\mathcal{C} \rightarrow \mathcal{C}$. All actegories in this paper will be symmetric.

An **enrichment** of a category $\mathcal{C}$ in a monoidal category $\mathcal{M}$ is a functor $[-, -] : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \mathcal{M}$ plus additional data and conditions. There is a tight connection between actegories and enrichments: if $\bullet$ is any actegory such that every $- \bullet X : \mathcal{M} \rightarrow \mathcal{C}$ has a right adjoint $[X, -] : \mathcal{C} \rightarrow \mathcal{M}$ (called a closed actegory) then $[-, -]$ is an enrichment, and conversely if $[-, -]$ is an enrichment such that every $[X, -]$ has a left adjoint $- \bullet X$ (called a copowered or tensored enrichment) then $\bullet$ is an action. For example, if $\mathcal{C}$ is any category with all coproducts then it has a tensored enrichment in the cartesian monoidal category **Set** and therefore an action $\bullet : \mathbf{Set} \times \mathcal{C} \rightarrow \mathcal{C}$ given by $M \bullet X = \sum_M X$.

3.2 Parametrisation

Given an actegory $\bullet : \mathcal{M} \times \mathcal{C} \rightarrow \mathcal{C}$, a **parametrised morphism** $f : X \rightarrow Y$ in $\mathcal{C}$ is a pair of an object $M : \mathcal{M}$ and a morphism $f : M \bullet X \rightarrow Y$. The identity parametrised morphism is given by $I : \mathcal{M}$ and $\text{id}_X : I \bullet X \cong X \rightarrow X$. The composite of $(M, f : M \bullet X \rightarrow Y)$ and $(N, g : N \bullet Y \rightarrow Z)$ has parameter $N \otimes M$ and

morphism $(N \otimes M) \bullet X \xrightarrow{\cong} N \bullet (M \bullet X) \xrightarrow{N \bullet f} N \bullet Y \xrightarrow{g} Z$. A reparametrisation from $(M, f : M \bullet X \rightarrow Y)$ to $(N, g : N \bullet X \rightarrow Y)$ is a morphism $h : M \rightarrow N$ in C such that $f = g \circ (h \bullet X)$.

Given an actegory $\bullet : \mathcal{M} \times C \rightarrow C$, we have a bicategory whose objects are objects of C , 1-cells are parametrised morphisms and 2-cells are reparametrisations. This may be referred to by **Para** $_{\mathcal{M}}(C)$, **Para** $_{\bullet}(C)$ or simply **Para** (C) when unambiguous. We believe that when $\bullet$ is a symmetric monoidal actegory, **Para** $_{\bullet}(C)$ is a symmetric monoidal bicategory [HS19] but this has not yet been proven.

When we have an action $\bullet : \mathcal{M} \times C \rightarrow C$ and a symmetric lax monoidal functor $W : \mathcal{M} \rightarrow \mathbf{Set}$ (or sometimes **Cat**) with its cartesian product, and we extend $\bullet$ to an action of the category of elements $\int W$ by precomposing with the discrete fibration $\pi : \int W \rightarrow \mathcal{M}$ to obtain $(M, w) \bullet X = M \bullet X$. When W is lax monoidal with laxator $\nabla : W(M) \times W(N) \rightarrow W(M \otimes N)$, $\int W$ gains a symmetric monoidal product $(M, w_M) \otimes (N, w_N) = (M \otimes N, w_M \nabla w_N)$. We write **Para** $_{\mathcal{M}}^W(C)$ for **Para** $_{\int W}(C)$, and call this **weighted parametrisation** [Gav24].

Dually, a **coparametrised morphism** $f : X \rightarrow Y$ is a pair of an object $M : \mathcal{M}$ and a morphism $f : X \rightarrow M \bullet Y$. There is a category **CoPara** $_{\bullet}(C)$ of objects, coparametrised morphisms and reparametrisations.

Given a category C enriched in a monoidal category $\mathcal{M}$, an **externally parametrised morphism** $f : X \rightarrow Y$ of C is a pair of an object M of $\mathcal{M}$ and a morphism $f : M \rightarrow [X, Y]$ of $\mathcal{M}$ [Smi23]. There is once again a bicategory **Para** $_{\mathcal{M}}(C)$ of externally parametrised morphisms. In the case of a tensored enrichment this bicategory is equivalent to the previous one, but there are also interesting cases when they differ. Coparametrised morphisms cannot be defined for an enrichment that is not tensored.

3.3 Optics

Given a monoidal category $\mathcal{M}$ acting on categories C and $\mathcal{D}$, and given objects X, Y of C and X', Y' of $\mathcal{D}$, a **mixed optic** $\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)$ is an equivalence class of triples of an object $M : \mathcal{M}$, a coparametrised morphism $f : X \rightarrow M \bullet Y$ of C and a parametrised morphism $f' : M \bullet Y' \rightarrow X'$ of $\mathcal{D}$. The equivalence classes are generated by reparametrisations and satisfy the universal property of a coend, $\int^{M:\mathcal{M}} C(X, M \bullet Y) \times \mathcal{D}(M \bullet Y', X')$. There are two different string diagram notations for an optic (figure 1). The first considers them as morphisms of a monoidal category, composing left-to-right, with causality flowing clockwise from top-left. The second considers them as colours of an operad, composing outside-in, with causality flowing left-to-right.

There is a category **Optic** $_{\mathcal{M}}(C, \mathcal{D})$ whose objects are pairs and whose morphisms are optics. When both actions are symmetric, or equivalently are defined by a span of symmetric monoidal functors $C \leftarrow \mathcal{M} \rightarrow \mathcal{D}$, then **Optic** $_{\mathcal{M}}(C, \mathcal{D})$ is a symmetric monoidal category, with the tensor product on objects being pairwise.

In the common case that $\mathcal{M} = C = \mathcal{D}$ acts on itself by monoidal product, we write **Optic** (C) . The tensor product of **Optic** (C) is pairwise monoidal product. When the monoidal unit of C is terminal (which includes all Markov categories) then we have natural isomorphisms **Optic** $(C) \left(I, \left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \right) \cong C(I, X)$ and **Optic** $(C) \left(\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right), I \right) \cong C(X, X')$. We call morphisms in latter case **continuations**, and define the representable functor $\mathbb{K} = \mathbf{Optic}(C)(-, I) : \mathbf{Optic}(C)^{\text{op}} \rightarrow \mathbf{Set}$.

There are two common cases when the coend in the definition of optics can be eliminated using the ninja Yoneda lemma [Ril18, Lor21]. Firstly, when $\mathcal{M} = C = \mathcal{D}$ acts on itself by cartesian product then there is a natural isomorphism **Optic** $\left(\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right), \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)\right) \cong C(X, Y) \times C(X \times Y', X')$. This is usually known as

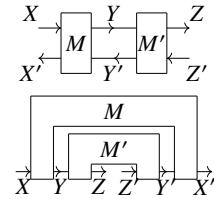


Figure 1: Alternative notations for optic composition

a **Lens**. Although this case is much easier to understand, there are significant conceptual advantages to the more general definition [Gav22]. Secondly, when $\mathcal{M} = \mathcal{C} = \mathcal{D}$ acts on itself by a closed monoidal product then there is a natural isomorphism $\mathbf{Optic}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}, \begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right) \cong \mathcal{C}(X, Y \otimes [Y', X'])$. Both of these cases can be generalised to requiring a condition on only one side.

For the cartesian self-action of **Set**, $\mathbf{Optic}(\mathbf{Set})$ coincides with the category of monomial endofunctors (those of the form $F(X) = A \times X^B$) and natural transformations. Any cartesian self-action in a category with finite limits can be generalised to **dependent lenses** (also known as morphisms of **containers** [AAG03]), which in the locally cartesian closed case are equivalent to **polynomial endofunctors** [NS24]. Finding the best of both worlds between the monoidal and cartesian cases is known as **dependent optics** [Ver23] and is only partially understood. There are reasons to want to use dependent optics in this paper because it is common that the available actions of a reinforcement learning agent depends on the current state of the Markov chain [BJI⁺17], but we only consider the simply-typed case in this paper for simplicity.

3.4 Parametrised optics

When a category of optics is symmetric monoidal, it admits a self-action. In [CGHR22] it was identified that the resulting category $\mathbf{Para}(\mathbf{Optic})$ of **parametrised optics** is extremely rich, and provides a general-purpose foundation for the study of controlled processes. The study of this is known as **categorical cybernetics**, which includes compositional game theory [Dal19], deep learning [CGG⁺22], compositional Bayesian inference [BHS23] and variational learning [Smi23], and applications in software engineering such as *open servers* [VC22].

4 States, contexts and iteration

An optic (whether parametrised or not) is a process consisting of a forward pass followed by a backward pass. In many applications, including those in this paper, this process is iterated through repeated interaction with an outside environment. In the case of supervised learning, this could simply be samples drawn from a dataset. In this section we will develop a general theory of iterated optics.

Let $\mathcal{C}$ be a symmetric monoidal category and $W : \mathcal{C} \rightarrow \mathbf{Set}$ a symmetric lax monoidal functor. Consider the bicategory $\mathbf{Para}^W(\mathcal{C})$ generated by the action of $\int W$ on $\mathcal{C}$ given by $(M, w) \bullet X = M \otimes X$. A parametrised morphism $X \rightarrow Y$ of $\mathcal{C}$ weighted by W consists of a morphism $M \otimes X \rightarrow Y$ together with an element $w \in W(M)$, as depicted in figure 2(right).

Any bicategory can be turned into a 1-category by change of enrichment basis along the connected components functor $\pi_0 : \mathbf{Cat} \rightarrow \mathbf{Set}$. This operation quotients together 1-cells that are related by any 2-cell. (π_0 is right adjoint to the free functor $\mathbf{Set} \rightarrow \mathbf{Cat}$, and it is more common to change basis along the left adjoint, which instead quotients out only invertible 2-cells.) The 1-category $\pi_0^*(\mathbf{Para}^W(\mathcal{C}))$ has morphisms that are equivalence classes identifying all ways of making the cut in figure 2(right). This satisfies an important universal property: it is the symmetric monoidal category that results from freely extending $\mathcal{C}$ with a state $w : I \rightarrow X$ for each element $w \in W(X)$, for all objects X [HT12]².

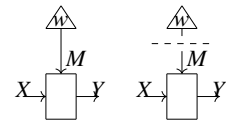


Figure 2: Morphism in $\mathbf{Para}^W(\mathcal{C})$ (right) and its equivalence class in $\pi_0^*(\mathbf{Para}^W(\mathcal{C}))$ (left).

²Thanks to Nathan Corbyn for bringing this reference to our attention.

Let C be a symmetric monoidal category. We define a symmetric lax monoidal functor called the **iteration functor**, $\mathbb{I} : \mathbf{Optic}(C) \rightarrow \mathbf{Set}$ [Hed24]. On objects, we set

$$\mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) = \int^{M:C} C(I, M \otimes X) \times C(M \otimes X', M \otimes X)$$

Given a representative element $(M, x_0, i) \in \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$ we call M the **state space**, $x_0 : I \rightarrow M \otimes X$ the **initial state** and $i : M \otimes X' \rightarrow M \otimes X$ the **iterator**.

Given an optic $f = (N, f, f') : \left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)$ in $\mathbf{Optic}(C)$, we get a function $\mathbb{I}(f) : \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow \mathbb{I}\left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)$ given by taking (M, x_0, i) to the state space $M \otimes N$, the initial state $I \xrightarrow{x_0} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y$, and the iterator

$$M \otimes N \otimes Y' \xrightarrow{M \otimes f'} M \otimes X' \xrightarrow{i} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y$$

This can be easily checked to be functorial and well-defined (see Appendix A for proofs).

Proposition 4.1. *The iterator $\mathbb{I} : \mathbf{Optic}(C) \rightarrow \mathbf{Set}$ is well-defined.*

Proposition 4.2. *The iterator $\mathbb{I} : \mathbf{Optic}(C) \rightarrow \mathbf{Set}$ is functorial.*

When $C = \mathbf{Set}$ and similar cases, given an element $i = (M, (m_0, x_0), i) \in \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$ and a function $k : X \rightarrow X'$, we can define an infinite sequence $\langle k|i \rangle : X^\omega$ by the corecursive formula

$$\langle k|M, (m_0, x_0), i \rangle = x_0 : \langle k|M, i(m_0, k(x_0)), i \rangle$$

This defines a dinatural transformation $\langle - | - \rangle : \mathbb{K}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \times \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow X^\omega$ which is well-defined.

Proposition 4.3. *The map $\langle - | - \rangle : \mathbb{K}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \times \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow X^\omega$ is a dinatural transformation when $C = \mathbf{Set}$.*

In the general case, we believe this can be accomplished using the machinery of monoidal streams [LFR22].

We also have an evident laxator $\nabla : \mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \times \mathbb{I}\left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right) \rightarrow \mathbb{I}\left(\begin{smallmatrix} X \otimes Y \\ X' \otimes Y' \end{smallmatrix}\right)$ defined up to symmetries by

$$(M, x_0, i) \nabla (M', x'_0, i') = (M \otimes M', x_0 \otimes x'_0, i_0 \otimes i'_0)$$

The resulting symmetric monoidal category $\mathbf{Optic}^{\mathbb{I}}(C) = \pi_0^*(\mathbf{Para}^{\mathbb{I}}(\mathbf{Optic}(C)))$ extends $\mathbf{Optic}(C)$ with states $I \rightarrow \left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$ defined by elements of $\mathbb{I}\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$. A typical morphism is depicted in figure 10.

Given a symmetric monoidal category C , a **comorphism** $X \rightarrow X'$, which could also be called a **context** for morphisms $X \rightarrow X'$, is a state $I \rightarrow \left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$ of $\mathbf{Optic}(C)$. When C is itself a category of optics, this is known as **double optics** and is a central idea of Bayesian open games [BHZ23]. These can be depicted as combs with 1 hole and bidirectional wires, or combs with 2 holes and only forwards wires.

Given a symmetric monoidal category C , a functor $\mathbf{Optic}(C) \rightarrow \mathbf{Set}$ can be equivalently defined as a **Tambara comodule**: a functor $W : C \times C^{\text{op}} \rightarrow \mathbf{Set}$ (note the variance) equipped with a natural family of functions $W(M \otimes X, M \otimes Y) \rightarrow W(X, Y)$. This is a dualisation of the fundamental theorem of Tambara theory [PS08, CEG⁺24]. Given this data, a generalised comorphism $X \rightarrow X'$ can be defined as an element of $W\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right)$, that is, a state of $\mathbf{Optic}^W(C) = \pi_0^*(\mathbf{Para}^W(\mathbf{Optic}(C)))$. This construction appears in Sec.9 of [Hed23].³

Putting this together, we can define an **iteration context** for optics $\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)$ as a (representable) state of $\mathbf{Optic}(\mathbf{Optic}^{\mathbb{I}}(C))$. This defines a functor $\mathbb{I}_{\text{env}} : \mathbf{Optic}^2(C) \rightarrow \mathbf{Set}$ depicted in figure 3(top), and

³The first author has been working on the theory of generalised contexts for several years, but has yet to find a compelling application outside of categorical cybernetics.

by pulling the state variable of i through k we can define it equivalently by a coend over C rather than over $\mathbf{Optic}^I(C)$:

$$\mathbb{I}_{\text{env}}\left(\left(\begin{smallmatrix} X \\ X' \end{smallmatrix}\right), \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix}\right)\right) \cong \int^{M, M' : C} C(I, M \otimes X) \times C(M \otimes Y, M' \otimes Y') \times C(M' \otimes X', M \otimes X)$$

This can be equivalently depicted as a 3-hole comb in figure 3(bottom), and we can unroll this n steps to produce a $2n$ -hole comb, including an ω -comb [LFR22] for the limiting case.

5 Bellman operators

A *Bellman operator* is a self-mapping of a function space of either state-value functions or state-action-value functions, which iteratively improves the estimation of values. For dynamic programming, the most basic Bellman operator $\mathfrak{B}_\pi = \mathfrak{B}_{\text{val}}(-, \pi) : \mathbb{R}^S \rightarrow \mathbb{R}^S$ is defined by

$$\mathfrak{B}_\pi(V)(s) = \mathbb{E}_{(r, s') \sim t(s, \pi(s))} [r + \gamma V(s')]$$

The functional equation $V = \mathfrak{B}_\pi(V)$ is called a *Bellman equation*, and its solution V characterises the long-run values of the policy π . Provided S is finite and $0 < \gamma < 1$, $\mathfrak{B}_\pi$ is a contraction mapping on the supremum metric of $\mathbb{R}^S$, and therefore iterating $\mathfrak{B}_\pi$ from any initial estimate of V will converge to the unique solution of the Bellman equation.

In [HS23] we showed that $\mathfrak{B}_\pi$ has the form $\mathfrak{B}_\pi = \mathbb{K}(\ell_\pi)$, where $\ell_\pi = \begin{pmatrix} \ell_f \\ \ell_b \end{pmatrix} : \begin{pmatrix} S \\ \mathbb{R} \end{pmatrix} \rightarrow \begin{pmatrix} S \\ \mathbb{R} \end{pmatrix}$ is the mixed optic in the category $\mathbf{Optic}_{\mathbf{Kl}(D)}(\mathbf{Kl}(D), \mathbf{EM}(D))$ defined by $\ell_f : S \rightarrow D(S \times \mathbb{R})$ which maps $s \mapsto t(s, \pi(s))$ and $\ell_b : D(\mathbb{R}) \times \mathbb{R} \rightarrow \mathbb{R}$ which maps $r, v \mapsto \mathbb{E}[r] + \gamma v$. Here D is the finite support probability monad on $\mathbf{Set}$, whose Eilenberg-Moore category is convex sets [Fri09], with $\mathbf{Kl}(D)$ acting on $\mathbf{EM}(D)$ via the embedding of free algebras as algebras $D : \mathbf{Kl}(D) \hookrightarrow \mathbf{EM}(D)$, which is a strong monoidal functor.

That is to say, if we represent a value function $V : S \rightarrow \mathbb{R}$ by a costate of optics $V : \begin{pmatrix} S \\ \mathbb{R} \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, then the costate $V \circ \mathfrak{B}_\pi : \begin{pmatrix} S \\ \mathbb{R} \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ similarly represents $\mathfrak{B}(V) : S \rightarrow \mathbb{R}$. This is a refinement of the usual view of Bellman operators as endomorphisms of function spaces.

Strictly speaking it would be preferable to use a category of optics whose forward objects are finite sets and backward objects are complete metric spaces, using a suitable probability monad such as Kantorovich [FP19], in order to guarantee convergence in all cases. However we leave the details of this for future work. For the general theory of RL one can work with Markov categories [Fri20], and especially representable Markov categories [FGPR23] to handle the interplay between *distributions* and *samples*.

This type of Bellman operator updates an entire value function at once, as is most common in basic dynamic programming. However in reinforcement learning it is far more common to update a Q-matrix one entry at a time, with a sample determining which state-action pair is to have its value updated. Bellman operators of that form do not directly factor through $\mathbb{K}$. However we can fix this with a small change in perspective: we consider Bellman operators that return a *delta* or *change* to a Q-matrix, apply $\mathbb{K}$ to that, and then apply the change to obtain a new Q-matrix in the world of functions rather than optics. This same change of perspective is required anyway to describe deep RL where the delta is replaced with a cotangent vector, so it is an interesting observation that the category theory suggests the same for the more discrete setting of tabular RL.

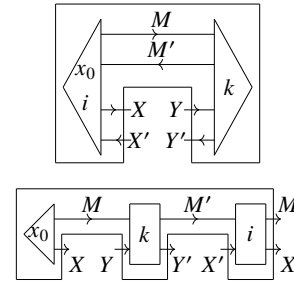


Figure 3: Iteration contexts as states of $\mathbf{Optic}^2(C)$ (top) and 3-hole combs (bottom).

One may ask whether $\mathfrak{B}_{\text{pol}}$ arises also as the image under $\mathbb{K}$ of a some optic, and the answer is negative. This was an ambiguous point in [HS23] where the policy improvement step could not be stated as an optic, even though both $\mathfrak{B}_{\text{val}}$ and $\mathfrak{B}_{\text{pol}}$ are treated as similar contraction operators in the dynamic programming literature. We can now give a more explicit answer; $\mathfrak{B}_{\text{pol}}$ involves two currying maps (λ), so it is not in the image of an optic under the continuation functor (see figure 4).

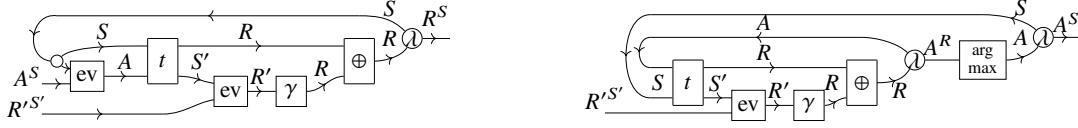


Figure 4: $\mathfrak{B}_{\text{val}}$ (left) and $\mathfrak{B}_{\text{pol}}$ (right) in the free autonomisation of **Set** [Del20] (see also remark 5.1).

Remark 5.1. The free autonomisation of $(\mathbf{Set}, \times, 1)$, denoted $(L\mathbf{Set}, \otimes, I)$, consists of formal string diagrams annotated with functions in **Set** [Del20]. The autonomous structure for the monoidal product $\otimes$ and the cartesian structure for the cartesian product $\times$ being different prevents $L\mathbf{Set}$ from having all objects be isomorphic to the monoidal unit. Forward morphisms are embeddings via the strong monoidal fully faithful functor $F : \mathbf{Set} \rightarrow L\mathbf{Set}$. Cups $\varepsilon : A \otimes A^* \rightarrow I$ and caps $\eta : I \rightarrow A^* \otimes A$ are freely added, and allow to draw dual objects A^* as backward wires. The diagrammatic notation for $L\mathbf{Set}$ in Figure 4 uses the isomorphism $\lambda : A^* \otimes B \cong B^A$ and the cap $\eta : I \rightarrow A^* \otimes A$. Morphisms in **Set** like $\text{ev} : B^A \times A \rightarrow B$ embed again via F .

5.1 Parametric Bellman operators

Since the category $\mathbf{Lens} = \mathbf{Optic}(\mathbf{Set})$ is enriched in **Set**, we can form its category $\mathbf{ParaSet}(\mathbf{Lens})$ of externally parametrised morphisms. A morphism $\binom{X}{X'} \rightarrow \binom{Y}{Y'}$ of this category consists of a parameter set P , a forwards pass function $P \times X \rightarrow Y$ and a backwards pass function $P \times X \times Y' \rightarrow X'$.

The simplest Bellman operator for RL, the one for SARSA, is a morphism in this category of type $\mathfrak{B} : \binom{1}{S \times A \times \mathbb{R}} \rightarrow \binom{S \times A}{\mathbb{R}}$, with parameter set $\Upsilon = S \times A \times \mathbb{R} \times S \times A$, where the backward pass function is $r, v \mapsto r + \gamma v$ (figure 5).

We lift the functor $\mathbb{K} : \mathbf{Lens}^{\text{op}} \rightarrow \mathbf{Set}$ to the functor $\mathbf{ParaSet}(\mathbb{K}) : \mathbf{ParaSet}(\mathbf{Lens}^{\text{op}}) \rightarrow \mathbf{ParaSet}(\mathbf{Set})$. Applying this functor to $\mathfrak{B}$ results in a function

$$\begin{aligned} \mathbf{ParaSet}(\mathbb{K})(\mathfrak{B}) : \Upsilon \times \mathbb{R}^{S \times A} &\rightarrow S \times A \times \mathbb{R} \\ ((s, a, r, s', a'), Q) &\mapsto (s, a, r + \gamma Q(s', a')) \end{aligned}$$

Here we very informally think of $S \times A \times \mathbb{R}$ as a ‘discrete cotangent vector’ at $Q \in \mathbb{R}^{S \times A}$. Slightly more precisely, the actual cotangent space is $\mathbb{R}^{S \times A}$, with $S \times A \times \mathbb{R}$ representing a scaled basis vector via the embedding $S \times A \times \mathbb{R} \hookrightarrow \mathbb{R}^{S \times A}$, $(s, a, r) \mapsto (s', a') \mapsto \begin{cases} r & \text{if } s = s', a = a' \\ 0 & \text{otherwise} \end{cases}$. This differential geometry per-

spective is heavily inspired by Myers’ categorical systems theory [Mye23] and related unpublished work in progress of Capucci, and we leave it for future work to make its application to reinforcement learning precise.

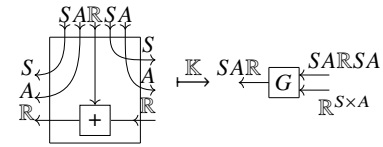


Figure 5: Target computation in SARSA as a parametrised lens, and its $\mathbb{K}$ -image in **Set**.

We do not provide a general definition of Bellman operators, and consider this a representative ‘definition by example’. In general, Bellman operators will be optics from value functions to deltas of value functions, parametrised by a *sample* which is data received from the environment.

In methods where the sample requires the operator to make use of the continuation only once, this parametric operator can be represented as a lens. SARSA is an example of this, where the usage of Q by the target is *linear* in the sense of linear type theory. Setting aside the convergence properties of the Bellman operator, we treat it from this point on as a morphism $G : Y \times \mathbb{R}^S \rightarrow \mathbb{R}^S$ in **Set**. This morphism becomes a central part of our formalization of an RL model, explained next.

6 Models, agents and environments

A model for an RL method contains the data to generate the policy from certain inner parameters and the data to update those parameters based on bootstrapping and/or samples. It matches the structure of a lens from model parameters to agent interface, which we annotate in figure 6(a).

The forward map uses parameters of the method to generate a policy for the agent’s interaction with the environment. In Q-learning for example, which is a TD method (figure 6(b)), $P : \mathbb{R}^{S \times A} \rightarrow (\mathcal{P}A)^S$ takes the current Q-table $Q : \mathbb{R}^{S \times A}$ and returns the greedy policy $\pi : (\mathcal{P}A)^S$ defined by $\pi(s) = \operatorname{argmax}_a Q(s, a)$.

The backward map takes the return from the agent and the current parameters to generate an update target as a (often discrete) cotangent vector for the parameters. The return usually takes the form of some product of types S (states), A (actions) and $\mathbb{R}$ (rewards). In SARSA (figure 6(b)), G takes a sample (s, a, r, s', a') (right input) and the bootstrapped Q-table (upper input) to calculate the target $r + \gamma Q(s', a')$, which is the direction of the cotangent vector at (s, a) . In DP (figure 6(d)), there is no return from the agent, and the update target is the output of the Bellman operator $\mathfrak{B}_{\text{val}}$ as a section of the cotangent bundle: for every state $s \in S$, $\mathfrak{B}_{\text{val}}(V)(s)$ defines the direction that $V(s)$ must change to.

Certain DP methods like GPI or asynchronous DP [Ber82] benefit from treating the two Bellman operators as separate backward morphisms (figure 6(e)).

This also captures some deep reinforcement learning methods, which use a gradient-based approximation of the value functions instead of a tabular approach. An example is actor-critic methods [KT01]. We will not give a complete motivation of their definition, but in short, the traditional policy is a neural network called the *actor*, and is complemented by an additional function called the *critic* that learns a baseline value function. The actor $P : \Theta \rightarrow (S \rightarrow DA)$ and the critic $V : \Omega \rightarrow (S \rightarrow \mathbb{R})$ appear in the forward map of the diagram (figure 6(f)), but only the actor outputs to the right interface, as the critic does not act on the environment. Being a deep RL method, the actor map $P : \theta \mapsto \pi_\theta$ is a neural network, and the associated backward loss map $\mathcal{L}_{\text{ac}} : \Theta \times \Omega \times \text{SARS} \rightarrow \Delta\Theta$ is defined by the improvement of expected return $(\theta, \omega, s, a, r, -) \mapsto (r - V_\omega(s)) \nabla_\theta \log \pi_\theta(s, a)$ with the baseline given by V_ω . The critic map $V : \omega \mapsto V_\omega$ has as the backward loss map $\mathcal{L}_{\text{cr}} : \Omega \times \text{SARS} \rightarrow \Delta\Omega$ the reduction of policy update variance $(\omega, s, a, r, s') \mapsto r + \gamma V_\omega(s') - V_\omega(s)$.

This generic model lens, and any of the variants above, embed into **Optic**(C) which is extended to **Optic** ^{$\mathbb{I}$} (C) by an iteration functor $\mathbb{I}$ defined next. The left interface to the model optic is closed by two pieces of data: An initial state $q_0 : I \rightarrow M \otimes \Theta$ and an update rule $i : M \otimes \Delta\Theta \rightarrow M \otimes \Theta$ that acts as the iterator. The bootstrapping type M is usually the whole parameter space Θ , but the optic iteration functor allows M to be any state space, e.g. a subset of Θ or an alternative encoding of it. In gradient-free methods, the update is generally pointwise as in (2). Conversely, gradient-based methods use neural network optimizers like stochastic gradient descent, Adam and other variations as the update

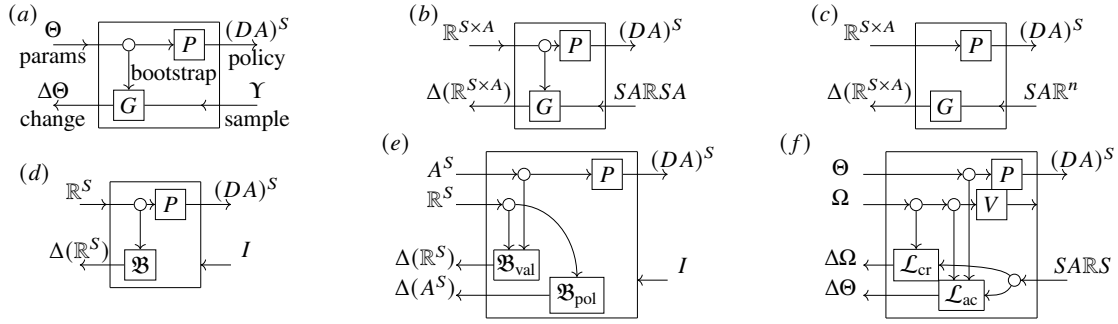


Figure 6: Lenses for (a) a generic RL model, (b) TD, (c) MC, (d) DP, (e) GPI and (f) Actor-Critic methods. The drawing of the backward map as a morphism with an input from the top is merely a stylistic choice, where it should be understood as a morphism with two inputs, the bootstrap and the sample.

rule [CGG⁺22].

The right interface of a model, which is a morphism $(\begin{smallmatrix} S \\ I \end{smallmatrix}) \rightarrow (\begin{smallmatrix} A \\ F \end{smallmatrix})$ in $\mathbf{Para}(\mathbf{Optic}(C))$, parametrises an agent. This parametrised morphism will itself interact with an environment that is an iteration context. The coend in the environment is taken over states of the Markov chain.

Offline methods, unlike online methods, interact with the agent only in a trivial way by showing it experiential samples. This is shown by the types $M = S \times A \times F$ and $M' = I$, by which the continuation ignores the agent's action and just projects the action and feedback as a response to the agent. Moreover, the iterator type $C(M' \otimes I, M \otimes S) \cong C(I, M \otimes S)$ coincides with the initial state, which reflects the fact that the environment samples experiences (s, a, f) from a distribution defined by a dataset (figure 8(a,b)).

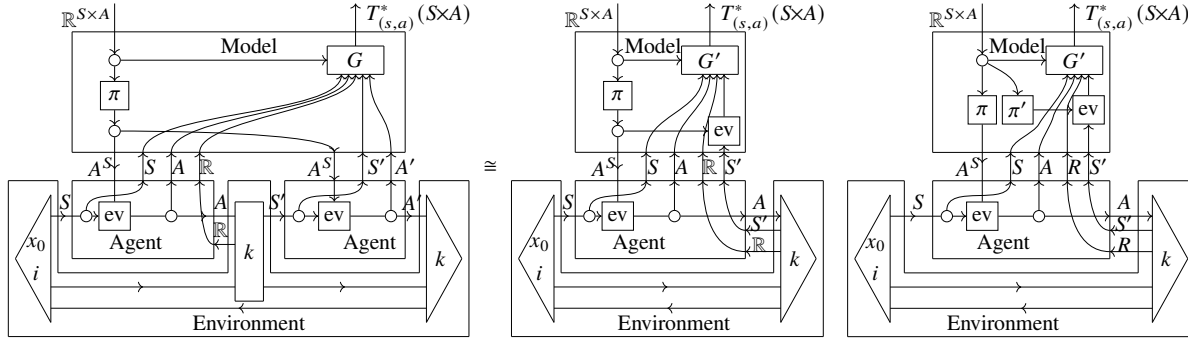


Figure 7: SARSA is on-policy (left two). Q-learning is off-policy (right).

To clarify the interplay between these the three structures described in this section, we look at the role played by internal and external policies in on- and off-policy methods. First, figure 7(left) shows the full representation of SARSA. It consists of a model optic parametrising two copies of an agent that are composed with a 2-hole environment. The policy evaluated by both instances is the same, and the return to the model consists of (s, a, r) from the first agent optic and (s', a') from the second. SARSA is an on-policy method, as the policy deployed to obtain $a' \sim \pi(s')$ is the same as the one used to compute the first action $a \sim \pi(s)$. Calculating the target G from the sample (s, a, r, s', a') is equivalent to calculating G from (s, a, r, s') and its internal policy π , even though the model does not know the environment's dynamics k . This is why the same method can be equivalently specified by the middle diagram.

On the other hand, Q-learning (figure 7(right)) is an off-policy method, because the last action is

computed by an internal policy $\pi' = \operatorname{argmax}$ different from the one being deployed.

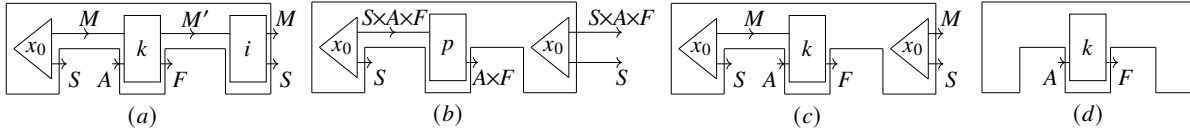


Figure 8: Online (a) and offline (b) RL environments. Contextual (c) and multi-armed (d) bandit environments. Omitted arrows are the unit. The offline continuation is a projection p of $A \times F$.

6.1 Prediction and bandit problems

The presented framework handles RL **prediction** problems for free in all the previous methods by trivialising the set $A = 1$, which pinpoints the idea that *a prediction algorithm is a control algorithm where there's no choice of actions*. For example, MC prediction of the long-term value of states from n -long episodes becomes an optic $(\mathbb{R}^S_{T^*_S}) \rightarrow (\mathbb{I}_{S^{\mathbb{R}^n}})$, and 1-TD prediction becomes $(\mathbb{R}^S_{T^*_S}) \rightarrow (\mathbb{I}_{S^{\mathbb{R}^S}})$. The forward maps for both are trivial since the agent has no policy to execute, perhaps better called **observer** rather than agent here. The corresponding environments have the type of a MRP.

Moreover, **bandit problems** emerge by trivialising $M' = I$ (figure 8(c,d)). In particular, contextual bandits involve finding the best action in A associated to a particular state in M for which only partial information of type S is given, yielding feedback in F . This action does not affect further distributions of states, so the object between the continuation and the update rule is trivial. Multi-armed bandit problems are a further special case, characterized by environments whose only non-trivial morphism is the continuation $k : A \rightarrow F$.

References

- [AAG03] Michael Abbott, Thorsten Altenkirch & Neil Ghani (2003): *Categories of containers*. In: *Proceedings of FoSSACS 2003*, Lecture Notes in Computer Science, Springer, doi:10.1007/3-540-36576-1_2.
- [Ber82] D. Bertsekas (1982): *Distributed dynamic programming*. *IEEE Transactions on Automatic Control* 27(3), pp. 610–616, doi:10.1109/tac.1982.1102980.
- [Ber19] Dimitri P. Bertsekas (2019): *Reinforcement Learning and Optimal Control*. Athena Scientific optimization and computation series, Athena Scientific.
- [BHS23] Dylan Braithwaite, Jules Hedges & Toby St Clere Smithe (2023): *The compositional structure of Bayesian inference*. In: *Proceedings of Mathematical Foundations of Computer Science 2023, Leibniz Proceedings in Informatics 272*, doi:10.48550/arXiv.2305.06112.
- [BHZ23] Joe Bolt, Jules Hedges & Philipp Zahn (2023): *Bayesian open games*. *Compositionality* 5(9), doi:10.32408/compositionality-5-9.
- [BJI⁺17] Nicola Botta, Patrik Jansson, Cezar Ionescu, David R. Christiansen & Edwin Brady (2017): *Sequential decision problems, dependent types and generic solutions*. *Logical Methods in Computer Science* 13(1), doi:10.23638/LMCS-13(1:7)2017.
- [BST22] Georgios Bakirtzis, Michail Savvas & Ufuk Topcu (2022): *Categorical semantics of compositional reinforcement learning*. arXiv:2208.13687.
- [CEG⁺24] Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore & Mario Román (2024): *Profunctor optics, a categorical update*. *Compositionality* 6(1), doi:10.32408/compositionality-6-1.

- [CG23] Matteo Capucci & Bruno Gavranović (2023): *Actegories for the working amthematician*. ArXiv:2203.16351.
- [CGG⁺22] Geoffrey Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2022): *Categorical foundations of gradient-based learning*. In: *Proceedings of ESOP 2022, Lecture Notes in Computer Science* 13240, doi:10.1007/978-3-030-99336-8_1.
- [CGHR22] Matteo Capucci, Bruno Gavranović, Jules Hedges & Eigil Rischel (2022): *Towards foundations of categorical cybernetics*. In: *Proceedings of Applied Category Theory 2021, Electronic Proceedings in Theoretical Computer Science* 372, doi:10.4204/EPTCS.372.17.
- [Dal19] Davidad Dalrymple (2019): *Dioptics: a common generalization of open games and gradient-based learners*. Unpublished paper available at <https://research.protocol.ai/publications/dioptics-a-common-generalization-of-open-games-and-gradient-based-learners/dalrymple2019.pdf>.
- [Del20] Antonin Delpeuch (2020): *Autonomization of Monoidal Categories*. In: *Proceedings of Applied Category Theory 2019, Electronic Proceedings in Theoretical Computer Science*, doi:10.4204/EPTCS.323.3.
- [FGPR23] Tobias Fritz, Tomáš Gonda, Paolo Perrone & Eigil Fjeldgren Rischel (2023): *Representable Markov categories and comparison of statistical experiments in categorical probability*. *Theoretical computer science* 961, doi:10.1016/j.tcs.2023.113896.
- [FP19] Tobias Fritz & Paolo Perrone (2019): *A probability monad as the colimit of spaces of finite samples*. *Theory and applications of categories* 34(7), pp. 170–220, doi:10.48550/arXiv.1712.05363. arXiv:1712.05363.
- [Fri09] Tobias Fritz (2009): *Convex spaces I: Definitions and examples*. ArXiv:0903.5522.
- [Fri20] Tobias Fritz (2020): *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics*. *Advances in Mathematics* 370, p. 107239, doi:10.1016/j.aim.2020.107239.
- [Gav22] Bruno Gavranović (2022): *Space-time tradeoffs of lenses and optics via higher category theory*. ArXiv: 2209.09351.
- [Gav24] Bruno Gavranović (2024): *Fundamental components of deep learning: A category-theoretic approach*. Ph.D. thesis, University of Strathclyde.
- [Ghr08] Robert Ghrist (2008): *Barcodes: The persistent topology of data*. *Bulletin of the American Mathematical Society* 45, doi:10.1090/S0273-0979-07-01191-3.
- [GHWZ18] Neil Ghani, Jules Hedges, Viktor Winschel & Philipp Zahn (2018): *Compositional game theory*. In: *Proceedings of Logic in Computer Science 2018, ACM*, doi:10.1145/3209108.3209165.
- [Hed23] Jules Hedges (2023): *The game semantics of game theory*. In: *Samson Abramsky on Logic and Structure in Computer Science and Beyond, Outstanding contributions to logic* 25, Springer, doi:10.1007/978-3-031-24117-8_11.
- [Hed24] Jules Hedges (2024): *Iteration with optics*. Blog post available at <https://cybercat.institute/2024/02/22/iteration-optics/>.
- [HS19] Linde Hansen & Mike Shulman (2019): *Constructing symmetric monoidal bicategories functorially*. ArXiv:1910.09240.
- [HS23] Jules Hedges & Riu Rodríguez Sakamoto (2023): *Value iteration is optic composition*. In: *Proceedings of Applied Category Theory 2022, Electronic Proceedings in Theoretical Computer Science* 380, doi:10.4204/EPTCS.380.24.
- [HT12] Claudio Hermida & Robert D Tennent (2012): *Monoidal indeterminates and categories of possible worlds*. *Theoretical computer science* 430, doi:10.1016/j.tcs.2012.01.001.
- [KLM96] L. P. Kaelbling, M. L. Littman & A. W. Moore (1996): *Reinforcement Learning: A Survey*. *Journal of Artificial Intelligence Research* 4, pp. 237–285, doi:10.1613/jair.301.

- [KT01] Vijay Konda & John Tsitsiklis (2001): *Actor-Critic Algorithms*. Society for Industrial and Applied Mathematics 42.
- [LFR22] Elena di Lavore, Giovanni de Felice & Mario Román (2022): *Monoidal streams for dataflow programming*. In: *Proceedings of Logic in Computer Science 2022*, ACM, doi:10.1145/3531130.3533365.
- [Lor21] Fosco Loregian (2021): *(Co)end calculus*. Cambridge University Press, doi:10.1017/9781108778657.
- [Mye23] David Jaz Myers (2023): *Categorical systems theory*. Draft book.
- [NS24] Nelson Niu & David Spivak (2024): *Polynomial functors: A mathematical theory of interaction*. Draft book.
- [PS08] Craig Pastro & Ross Street (2008): *Doubles for monoidal categories*. Theory and applications of categories 21(4), doi:10.48550/arXiv.0711.1859.
- [Ril18] Mitchell Riley (2018): *Categories of optics*. ArXiv:1809.00738.
- [Smi23] Toby St Clere Smithe (2023): *Mathematical foundations for a compositional account of the Bayesian brain*. Ph.D. thesis, University of Oxford.
- [SSPS21] David Silver, Satinder Singh, Doina Precup & Richard S. Sutton (2021): *Reward is enough*. Artificial Intelligence 299, p. 103535, doi:10.1016/j.artint.2021.103535.
- [SSS⁺17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel & Demis Hassabis (2017): *Mastering the game of Go without human knowledge*. Nature 550, doi:10.1038/nature24270.
- [Sut95] Richard S. Sutton (1995): *Generalization in reinforcement learning: successful examples using sparse coarse coding*. In: *Proceedings of the 8th International Conference on Neural Information Processing Systems, NIPS'95*, MIT Press, pp. 1038–1044, doi:10.5555/2998828.2998974.
- [VBC⁺19] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps & David Silver (2019): *Grandmaster level in StarCraft II using multi-agent reinforcement learning*. Nature 575, doi:10.1038/s41586-019-1724-z.
- [VC22] André Videla & Matteo Capucci (2022): *Lenses for composable servers*. ArXiv:2203.15633.
- [Ver23] Pietro Vertechhi (2023): *Dependent optics*. In: *Proceedings of Applied Category Theory 2022*, EPTCS, doi:10.4204/EPTCS.380.8.
- [WD92] Christopher J. C. H. Watkins & Peter Dayan (1992): *Q-learning*. Machine Learning 8(3-4), pp. 279–292, doi:10.1007/bf00992698.

A Appendix

Proof (4.1). We have to prove that the following function is well-defined:

$$\mathbf{Optic} \left(\left(\begin{smallmatrix} X \\ X' \end{smallmatrix} \right), \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix} \right) \right) \rightarrow \left[\mathbb{I} \left(\begin{smallmatrix} X \\ X' \end{smallmatrix} \right) \rightarrow \mathbb{I} \left(\begin{smallmatrix} Y \\ Y' \end{smallmatrix} \right) \right] \quad (3)$$

Following Riley's proof method of sequential composition of optics [Ril18], the uncurried form of the above function has as domain:

$$\begin{aligned} & \left(\int^M C(X, M \otimes Y) \times C(M \otimes Y', X') \right) \times \left(\int^N C(I, N \otimes X) \times C(N \otimes X', N \otimes X) \right) \\ & \cong \int^{M,N} C(X, M \otimes Y) \times C(M \otimes Y', X') \times C(I, N \otimes X) \times C(N \otimes X', N \otimes X) \quad (\text{coend-Fubini}) \end{aligned}$$

By the universal property of coends it suffices to construct maps natural in M and N into the codomain of (3):

$$\begin{aligned} & C(X, M \otimes Y) \times C(M \otimes Y', X') \times C(I, N \otimes X) \times C(N \otimes X', N \otimes X) \\ & \rightarrow C(I, N \otimes M \otimes Y) \times C(N \otimes M \otimes Y', N \otimes M \otimes X) \quad (\text{composition in } C) \\ & \rightarrow \int^P C(I, P \otimes X) \times C(P \otimes X', P \otimes X) \quad (\text{copr}_{N \otimes M}) \end{aligned}$$

where the first map takes morphisms f, f', x_0, i to

$$\begin{aligned} I & \xrightarrow{x_0} N \otimes X \xrightarrow{N \otimes f} N \otimes M \otimes Y \\ N \otimes M \otimes Y' & \xrightarrow{N \otimes f'} N \otimes X \xrightarrow{i} N \otimes X \xrightarrow{N \otimes f} N \otimes M \otimes Y \end{aligned}$$

This also admits a graphical representation, depicted in Figure 9. □

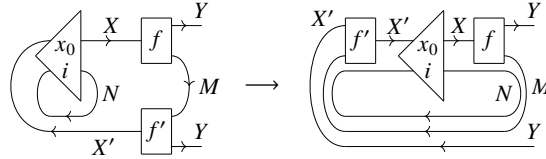


Figure 9: Composition of an optic with an iterator yields another iterator.

Proof (4.2). Let $f = (N, f, f') : \binom{X}{X'} \rightarrow \binom{Y}{Y'}$ and $g = (P, g, g') : \binom{Y}{Y'} \rightarrow \binom{Z}{Z'}$ be two morphisms in **Optic**(C). Preservation of identity is shown by:

$$\mathbb{I}(I, 1_X, 1_{X'}) : (M, x_0, i) \mapsto (M \otimes I, x_0; (I \otimes 1_X), (M \otimes 1_{X'}); i; (M \otimes 1_X))$$

Preservation of composition is shown by the isomorphic images of $\mathbb{I}(N, f, f'); \mathbb{I}(P, g, g')$, which maps $(M, x_0, i) : \mathbb{I}\left(\binom{X}{X'}\right)$ to the state space $M \otimes N \otimes P$, the initial state $I \xrightarrow{x_0} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y \xrightarrow{M \otimes N \otimes g} M \otimes N \otimes P \otimes Z$ and the iterator

$$M \otimes N \otimes P \otimes Z' \xrightarrow{M \otimes N \otimes g'} M \otimes N \otimes Y' \xrightarrow{M \otimes f'} M \otimes X' \xrightarrow{i} M \otimes X \xrightarrow{M \otimes f} M \otimes N \otimes Y \xrightarrow{M \otimes N \otimes g} M \otimes N \otimes P \otimes Z$$

which defines the element in $\mathbb{I}\left(\binom{Z}{Z'}\right)$, and $\mathbb{I}(N \otimes P, (f; N \otimes g), (N \otimes g'; f'))$, which maps (M, x_0, i) to the same state space, the initial state $x_0; (M \otimes (f; N \otimes g))$, and the iterator $(M \otimes (N \otimes g'); f'); i; (M \otimes (f; N \otimes g))$. □

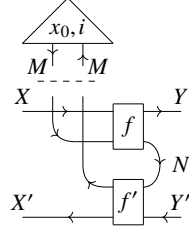


Figure 10: Typical morphism in $\mathbf{Optic}^{\mathbb{I}}(C) = \pi_0^*(\mathbf{Para}^{\mathbb{I}}(\mathbf{Optic}(C)))$ which consists of a morphism in $\mathbf{Optic}(C)$ extended by a representative element $(M, x_0, i) \in \mathbb{I}(\frac{X}{X})$. Adapted from [Hed24].

Proof (4.3). Considering that $\mathbf{Optic}(\mathbf{Set}) \cong \mathbf{Lens}$, the domain and codomain functors of the transformation are $\mathbb{K} \times \mathbb{I} : \mathbf{Lens}^{\text{op}} \times \mathbf{Lens} \rightarrow \mathbf{Set}$ and $\mathbb{V}^\omega : \mathbf{Lens} \rightarrow \mathbf{Set}$, where $\mathbb{V}^\omega$ is the forwards pass functor $\mathbb{V} : \mathbf{Lens} \rightarrow \mathbf{Set}$ followed by the stream functor $(-)^{\omega} : \mathbf{Set} \rightarrow \mathbf{Set}$. This being a purely covariant functor makes the dinaturality condition into the following pentagon identity for every lens $\lambda : (\frac{X}{X'}) \rightarrow (\frac{Y}{Y'})$ whose forward and backward maps we denote f and f' :

$$\begin{array}{ccc}
 & \mathbb{K}(\lambda) \times \mathbb{I}(\frac{X}{X'}) & \\
 \nearrow & & \searrow \\
 \mathbb{K}(\frac{Y}{Y'}) \times \mathbb{I}(\frac{X}{X'}) & & \mathbb{K}(\frac{Y}{Y'}) \times \mathbb{I}(\lambda) \\
 & \searrow & \nearrow \\
 & \mathbb{K}(\frac{X}{X'}) \times \mathbb{I}(\frac{X}{X'}) & \\
 & \xrightarrow{\langle - | - \rangle_{(\frac{X}{X'})}} \mathbb{V}^\omega(\frac{X}{X'}) = X^\omega & \\
 & \downarrow \mathbb{V}^\omega(\lambda) = f^\omega & \\
 & \mathbb{K}(\frac{Y}{Y'}) \times \mathbb{I}(\frac{Y}{Y'}) & \\
 & \xrightarrow{\langle - | - \rangle_{(\frac{Y}{Y'})}} \mathbb{V}^\omega(\frac{Y}{Y'}) = Y^\omega &
 \end{array}$$

For $k, (M, (m_0, x_0), i)$ in $\mathbb{K}(\frac{Y}{Y'}) \times \mathbb{I}(\frac{X}{X'})$, the diagram commutes when the streams $f^\omega \langle \lambda; k \mid M, (m_0, x_0), i \rangle$ and $\langle k \mid M, (m_0, f(x_0)), j \rangle$ are equal, where $j = (M \times f'); i; (M \times f)$. We proceed by coinduction, showing that the streams have equal heads and tails; an equivalent perspective is that they are generated by bisimilar (in fact isomorphic) state machines. The heads of both streams is $f(x_0)$, and their tails operate as state machines for states (m_0, x_0) :

$$\begin{aligned}
 f^\omega \langle \lambda; k \mid M, i(m_0, (\lambda; k)(x_0)), i \rangle &= f(i(m_0, (\lambda; k)(x_0)))_1 :: \langle \dots \rangle \\
 \langle k \mid M, j(m_0, k(f(x_0))), j \rangle &= j(m_0, k(f(x_0)))_1 :: \langle \dots \rangle
 \end{aligned}$$

Thus we show for all pairs (m, x) ,

$$\begin{aligned}
 & f(i(m, (\lambda; k)(x)))_1 \\
 &= f(i(m, (f; k; f')(x)))_1 && \text{(Composition of lens with continuation)} \\
 &= ((M \times f'); i; (M \times f))(m, k(f(x)))_1 \\
 &= j(m, k(f(x)))_1
 \end{aligned}$$

□

Organizing Physics with Open Energy-Driven Systems

Matteo Capucci

University of Strathclyde

matteo.capucci@gmail.com

Owen Lynch

Topos Institute

owen@topos.institute

David I. Spivak*

Topos Institute

david@topos.institute

Organizing physics has been a long-standing preoccupation of applied category theory, going back at least to Lawvere. We contribute to this research thread by noticing that Hamiltonian mechanics and gradient descent depend crucially on a consistent choice of transformation—which we call a reaction structure—from the cotangent bundle to the tangent bundle. We then construct a compositional theory of reaction structures. Reaction-based systems offer a different perspective on composition in physics than port-Hamiltonian systems or open classical mechanics, in that reaction-based composition does not create any new constraints that must be solved for algebraically.

The technical contributions of this paper are the development of symmetric monoidal categories of open energy-driven systems and open differential equations, and a functor between them, functioning as a “functorial semantics” for reaction structures. This approach echoes what has previously been done for open games and open gradient-based learners, and in fact subsumes the latter. We then illustrate our theory by constructing an n -fold pendulum as a composite of n -many pendula.

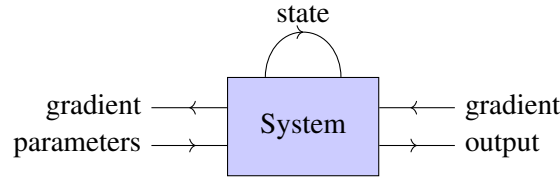
1 Introduction

A long-standing goal of applied category theory is to provide compositional frameworks for physical systems [Law80]. In this paper, we introduce a compositional framework for a generalization of both systems based on Hamiltonian mechanics and general systems which perform gradient descent. In this framework, systems interact by sending each other gradients. These gradients are converted to motion (that is, tangent vectors) by what we call a *reaction*. In Hamiltonian dynamics, the reaction arises from the symplectic structure and in gradient descent, the reaction arises from a Riemannian structure.

In Section 2 we explain the basic intuition: at the core, Hamiltonian dynamics and gradient descent on a manifold X both depend on a map $R: T^*X \rightarrow TX$, which translates gradients into motion. One can imagine this as a functional version of argmax, in that given a “valuation” function $\varphi: T_x X \rightarrow \mathbb{R}$, one obtains a choice of element $R(\varphi) \in T_x X$. But whereas gradient descent really does choose a tangent direction that maximizes change along the gradient, Hamiltonian dynamics does almost the opposite, choosing a tangent direction with zero change along the gradient.

Our compositional framework is bidirectional but oriented. This means that some parts of an open system are deemed inputs whereas other parts are deemed outputs. For example, in a pendulum, the input is the position and momentum of the pivot point, and the output is the position and momentum of the bob, which another system may take as input. However, the system also *receives* gradients on its output, and *sends* gradients to its input, which can be thought as the result of external forces. It also keeps and updates an internal state. This can be pictured in the following way.

*Spivak and Lynch’s contribution was supported by the Air Force Office of Scientific Research under award number FA9550-23-1-0376.



We begin in Section 2 by reviewing Hamiltonian and gradient descent systems and their commonalities, motivating the ensuing developments. Then we define symmetric monoidal category of open energy-driven systems in Section 3; there the bidirectionality is not explicit, but instead there is an explicit choice of reaction R . In Section 4 we define a functor that assigns these systems a bidirectional semantics in terms of parametric lenses. We do so in two steps, the first of which is more likely generalizable and the second of which is conceptually simpler. These semantics land in lens categories, which are now well-known in categorical machine learning literature. In fact our semantics is inspired by [FST19, CGG⁺22], which we extend to general smooth manifolds.

Finally, in Section 5 we give one example of these ideas, that of an n -fold pendulum constructed by composing single pendula. We conclude with an epilogue in Section 6.

1.1 Related work

This framework is connected to a variety of other attempts to formalize physical systems within applied category theory.

- The title of this paper is a pun based on the double category $\mathbb{O}rg$ that was first developed by the third-named author in [Spi22, SS23]. One way of thinking about this paper is that it develops a “continuous version of $\mathbb{O}rg$.”
- Another approach to “systems exerting force on each other” is port-Hamiltonian systems, which has been developed categorically by the second-named author in [Lyn22], [LLL24]. However, the doctrine of composition developed for port-Hamiltonian systems is undirected and relational, and so a computer implementation of this composition would require solving differential-algebraic equations in a similar way to [MGA⁺21] rather than just differential equations. In contrast, the framework in the current paper gives an “input-output” view on physical systems, and thus composition does not introduce new constraints that must be solved for. In future work, we hope to give an account of the relationship between the directed and undirected accounts of composition.
- Classical mechanics has been previously treated from a category theoretic viewpoint via spans of symplectic manifolds [BWY21]. However, like port-Hamiltonian systems, this approach is essentially relational, necessitating semantics in differential-algebraic equations.
- We heavily rely on the Para construction and its functoriality, as well as using roughly the same techniques of [CGHR22] for constructing our symmetric monoidal categories of open systems.
- The pattern of having a simple description of feedback systems that then gets compiled down to parametric lenses of some sort follows what [Cap23] has shown for open games. The analogy with gradient-based systems, formulated in terms of changes and valuations, already appears in *ibid*.
- Resource sharing machines are another method of composing dynamical systems which has been applied to dynamical systems for physics [LBPf22]. In contrast with the present approach, resource sharing machines do not derive their dynamics from potentials and forces; rather a resource sharing machine takes the vector field as primitive. However, we hope that in the future “resource-sharing composition” will be available to use with the formalism of this paper.

1.2 Notational conventions

For composition, we use $\circ$ to denote diagrammatic order. For a vector bundle $\pi: E \rightarrow B$, we denote the set of global sections by $\Gamma(\pi) := \{s: B \rightarrow E \mid s \circ \pi = 1\}$. Given another vector bundle $\pi': E' \rightarrow B$, we denote the vector bundle (over B) of fiberwise linear maps between them by $[\pi, \pi']$.

We denote by $(\mathbf{Mfd}, \mathbb{R}^0, \times)$ the cartesian monoidal category of smooth manifolds and smooth maps between them, although everything we're saying works in well-behaved generalizations, such as diffeological spaces [Sou06]. We denote the circle by $S^1 \in \mathbf{Mfd}$. The wide subcategory of smooth manifolds and isomorphisms between them is denoted $\mathbf{Mfd}^{\text{iso}}$; from which it inherits the (no longer cartesian) monoidal structure $(\mathbb{R}^0, \times)$.

1.3 Prerequisites

In order for the framework of this paper to be comprehensible, a certain amount of context must be given, but also a certain amount of context must be omitted for brevity.

The context that we omit and we assume the reader to be already acquainted with is basic differential geometry (definition and strong monoidal functoriality of tangent and cotangent bundles), and especially the theory of vector bundles (we reference [KMS93]). We also expect the reader to be familiar with the **Para** construction (see [FST19, CGHR22, CGG⁺22]) and its functorial properties, but we nonetheless spell out the result when we invoke it.

It will also help the reader to be familiar with Hamiltonian mechanics and gradient descent, but we will briefly review these so that a sufficiently determined reader may get through this paper without too much prior experience.

Lastly, while we kept this work strictly in the land of 1-categories for brevity, we use ideas from categorical systems theory and think of our constructions as shadows of their essentially double-categorical nature. So being aware of [Mye23a] is not required (except in passages where we explicitly draw a connection) but might help understanding the subtext.

2 Intuition

Hamiltonian mechanics and gradient descent have a common mathematical structure. We start by reviewing them.

2.1 Hamiltonian mechanics

A Hamiltonian system consists of a state space $X \in \mathbf{Mfd}$ with a full-rank symplectic form $\omega \in \Gamma(T^*X \wedge T^*X)$ and a function $H: X \rightarrow \mathbb{R}$, called the *Hamiltonian*, which represents the system's energy at any $x \in X$. Applying ω and the sequence of maps

$$T^*X \wedge T^*X \rightarrow T^*X \otimes T^*X \cong [TX, T^*X]$$

we produce a section $K_\omega \in \Gamma[TX, T^*X]$, i.e. a linear map $TX \rightarrow T^*X$ over X , which is invertible because ω is full-rank. Call its inverse $J_\omega \in \Gamma[T^*X, TX]$; we refer to it as the *reaction* associated to ω . Then the dynamics of the system (X, ω, H) are given by Hamilton's equation, i.e. the differential equation

$$\frac{dx}{dt}(t) = J_\omega(x(t)) \, dH(x(t)). \quad (1)$$

Note that the dynamics only depend on the reaction J_ω rather than ω ; we started with the symplectic form to connect to the more conventional way of doing things, but we will almost exclusively only work with reactions in the future.

Remark 1. Notice how we only need ω to be non-degenerate because we want to take the inverse of K_ω . However, if we had just started from J_ω , we need not assume that J_ω is invertible. In fact we get a reaction also from a Poisson structure [CFM21], which boils down to a possibly-degenerate J_ω . These can be used to model Hamiltonian systems with odd dimension.

For any smooth manifold $M \in \mathbf{Mfd}$ of dimension n , there is a canonical symplectic structure on its cotangent space $X := T^*M$. Given a coordinate chart for M and the induced coordinate chart on T^*M and TM , the corresponding reaction $J: T_x^*X \rightarrow T_xX$ over $x \in X$ has the following form

$$J(x) = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \quad (2)$$

where I_n is the $(n \times n)$ -identity matrix. Note that $J(x)$ is independent of the choice of coordinate chart [Arn89, Chapter 8].

Example 2. Consider the pendulum of fixed length l and mass m shown here



The little dot is called the *pivot* and the big dot is called the *bob*. We can model this as a Hamiltonian system by letting $X = T^*S^1$ with coordinates (θ, L) representing angle and angular momentum. Then (2) becomes

$$J(x) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

because S^1 is 1-dimensional. For a given map $H: T^*S^1 \rightarrow \mathbb{R}$, Eq. (1) reads

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial L}, \quad \frac{dL}{dt} = -\frac{\partial H}{\partial \theta}$$

To model a pendulum as in (3) with kinetic and gravitational energy, we would make the following Hamiltonian. First, we can compute several derived quantities from the variables (θ, L) of the system (the following quantities are all implicit functions of (θ, L)).

$$\begin{aligned} I &= ml^2 && \text{(moment of inertia)} \\ \omega &= \frac{L}{I} && \text{(rotational velocity)} \\ x &= l \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} && \text{(position of the mass)} \\ v &= l\omega \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix} && \text{(velocity of the mass)} \\ h &= x_2 = l \sin \theta && \text{(height of the mass)} \end{aligned}$$

The Hamiltonian is then the sum of kinetic and gravitational energy, written as

$$H(\theta, L) = \frac{1}{2}m\|v\|^2 + mgh$$

where g is gravitational acceleration. Plugging this H into Eq. (1) gives

$$\begin{aligned}\frac{d\theta}{dt} &= \frac{L}{I} \\ \frac{dL}{dt} &= -mgl \cos(\theta)\end{aligned}$$

Definition 3. For any manifold $X \in \mathbf{Mfd}$, a *reaction* on X is a map $J: T^*X \rightarrow TX$ over X . The set of reactions on X is denoted $\mathbf{React}(X)$. Given a smooth map $f: X \rightarrow Y$, we denote by $\mathbf{React}(f): \mathbf{React}(X) \rightarrow \mathbf{React}(Y)$ the map sending J to the composite $T^*Y \xrightarrow{T^*f} T^*X \xrightarrow{J} TX \xrightarrow{Tf} TY$.

2.2 Gradient descent

Gradient descent (or gradient ascent) is a very similar story. We start with a state space X along with a Riemmanian metric $g \in \Gamma(T^*X \otimes T^*X)$ and a function $S: X \rightarrow \mathbb{R}$. We apply the exact same procedure to g that we did to ω in order to get a section $M \in \mathbf{React}(X)$, and then we get the equation

$$v(x) = M(x) \, dS(x)$$

Example 4. Let $X = \mathbb{R}^2$ with the Euclidean metric. Then gradient ascent for a function $S: X \rightarrow \mathbb{R}$ looks like

$$\frac{dx_1}{dt} = \frac{\partial S}{\partial x_1}, \quad \frac{dx_2}{dt} = \frac{\partial S}{\partial x_2}$$

It is more traditional to write this as $\frac{dx}{dt} = \nabla S(x)^T$. It may seem like this doesn't use any fancy Riemannian structure, but in fact transposing a row vector into a column vector is made possible using the isomorphism $(\mathbb{R}^n)^* \cong \mathbb{R}^n$, the same isomorphism on which the natural inner product for $\mathbb{R}^n$ is built.

From Hamiltonian system and gradient descent systems we can provide the following common generalization:

Definition 5. An *energy-driven system* consists of a *state space* $X \in \mathbf{Mfd}$, a *reaction* $R \in \mathbf{React}(X)$, and an *energy functional* $E: X \rightarrow \mathbb{R}$.

The reaction embodies the “laws of physics”, turning energies (given by E) into forces.

3 Open Energy-driven Systems

Most, if not all, systems are in practice *open*, meaning they are amenable to composition with other systems.

Example 6. Take, for instance, the pendulum system from Example 2. In there we considered the pivot to be a fixed point, but in practice both pivot and bob are physical locations at which other systems can be attached. This means one can consider the pendulum as parametrized by $A = T\mathbb{R}^2$, the phase space of

the pivot, and to influence the location and velocity of the bob by exposing such quantities in $B = T\mathbb{R}^2$. Again, $X = T^*S^1$, but now, rather than $E: X \rightarrow \mathbb{R}$, we have $E: A \times X \rightarrow \mathbb{R}$ defined by

$$E((x_0, v_0), (\theta, L)) = \frac{1}{2m} \|v_0 + v\|^2 + mg(h_0 + h)$$

This E represents the potential plus kinetic energy of a pendulum whose pivot is at position x_0 and moving at velocity v_0 .

We then define $w: T\mathbb{R}^2 \times T^*S^1 \rightarrow T\mathbb{R}^2$ by

$$w((x_0, v_0), (\theta, L)) = (x_0 + x, v_0 + v)$$

This gives the position and velocity of the bob.

The “open pendulum” we just described is but an example of a general definition:

Definition 7. Given $A, B \in \mathbf{Mfd}$, an *open energy-driven system with inputs A and outputs B* consists of a manifold X , a reaction $R \in \Gamma[T^*X, TX]$, a function $E: A \times X \rightarrow \mathbb{R}$, and a function $w: A \times X \rightarrow B$. We call A the *inputs*, X the *state*, and B the *output*.

The fact that $A = B$ in Example 6 is tantalizing, because it makes the pendulum something like a “parametric endomorphism” of $T\mathbb{R}^2$. Could we somehow “compose the pendulum with itself” to make a double pendulum? Also, how can we describe the differential equation attached to an open energy-driven system; i.e. what is the semantics of an open energy-driven system? Answering these questions is the subject of the remainder of this paper.

3.1 Composing Open Energy-driven Systems

The purpose of this section is to develop a symmetric monoidal category where the morphisms are open energy-driven systems.

We begin with the following straightforward proposition.

Proposition 8. The functor $\mathbf{React}: \mathbf{Mfd}^{\text{iso}} \rightarrow \mathbf{Set}$ from Definition 3 is lax symmetric monoidal, with unitor $1 \rightarrow \mathbf{React}(1)$ given by the unique element of $\mathbf{React}(1)$ (the zero map $\mathbb{R}^0 \rightarrow \mathbb{R}^0$) and compositor

$$\oplus_{X_1, X_2}: \mathbf{React}(X_1) \times \mathbf{React}(X_2) \rightarrow \mathbf{React}(X_1 \times X_2) \quad (4)$$

defined by sending $R_1 \in \mathbf{React}(X_1), R_2 \in \mathbf{React}(X_2)$ to

$$T_{(x_1, x_2)}^*(X_1 \times X_2) \cong T_{x_1}^*X_1 \times T_{x_2}^*X_2 \xrightarrow{R_1(x_1) \times R_2(x_2)} T_{x_1}X_1 \times T_{x_2}X_2 \cong T_{(x_1, x_2)}(X_1 \times X_2).$$

Definition 9. Let $\int \mathbf{React} \xrightarrow{\pi_{\mathbf{React}}} \mathbf{Mfd}^{\text{iso}}$ be the monoidal Grothendieck construction of $\mathbf{React}$ (see [MV20]). An object of $\int \mathbf{React}$ is a pair (X, R) consisting of a space and a reaction on it, a morphism is a diffeomorphism which preserves the reactions, and the monoidal product $(X_1, R_1) \otimes (X_2, R_2)$ is given by $(X_1 \times X_2, R_1 \oplus R_2)$ as above.

Example 10. The symmetric monoidal category of symplectic manifolds and symplectomorphisms embeds faithfully into $\int \mathbf{React}$ by sending a symplectic manifold (X, ω) to (X, J_ω) , where J_ω is the reaction associated to ω , as constructed in Section 2.1. A *symplectomorphism* is a diffeomorphism that preserves the symplectic structure, and it is not too hard to show that preserving the symplectic structure implies preserving the reaction structure.

Now, as the composite $\int \mathbf{React} \xrightarrow{\pi_{\mathbf{React}}} \mathbf{Mfd}^{\text{iso}} \xrightarrow{U} \mathbf{Mfd}$ is strong monoidal, we have a monoidal action of $\int \mathbf{React}$ on $\mathbf{Mfd}$, simply given by $(X, R) \otimes_U Y := X \times Y$. The technique of decorating an action with data coming from a monoidal Grothendieck construction was introduced by [CGHR22].

We can then construct a category **OpenReact** by applying the Para construction [FST19, CGHR22] to $\otimes_U$, and in fact we use this opportunity to recall how the latter is performed.

Definition 11. Let **OpenReact** be the symmetric monoidal category of *open reaction systems* defined as the local 0-truncation¹ of $\mathbf{Para}(\otimes_U)$.

Concretely, an object in **OpenReact** is a manifold, a 1-cell $A \rightarrow B$ is an open reaction system and is given by a pair $(X, R) \in \int \mathbf{React}$ together with a smooth map $w: X \times A \rightarrow B$. These 1-cells compose by accumulating parameters if $(X_1, R_1, w_1: A \times X_1 \rightarrow B)$ and $(X_2, R_2, w_2: B \times X_2 \rightarrow C)$ are composable 1-cells, their composite is:

$$(X_1, R_1, w_1) \circ (X_2, R_2, w_2) := (X_1 \times X_2, R_1 \oplus R_2, A \times X_1 \times X_2 \xrightarrow{w_1 \times X_2} B \times X_2 \xrightarrow{w_2} C). \quad (5)$$

Morphisms (X, R, w) and (X', R', w') are considered equivalent if there is an isomorphism $i: X \cong X'$ with $\mathbf{React}(i)(R) = R'$ and $w = (i \times A) \circ w'$. The symmetric monoidal structure is the same as **Mfd** on objects, while on 1-cells $(X, R, w): A \rightarrow B$ and $(Y, S, v): C \rightarrow D$ is given by

$$(X, R, w) \otimes (Y, S, v) := (X \times Y, R \oplus S, (A \times C) \times (X \times Y) \xrightarrow{\sim} (X \times A) \times (Y \times C) \xrightarrow{w \times v} B \times D).$$

The fact that this symmetric monoidal structure is well-defined on the local 0-truncation is proven in [HT12].

Compared to energy-driven systems, reaction systems lack the data of an energy functional. We add this by considering such a functional as an effect. In fact $(\mathbb{R}, 0, +)$ is a monoid in **OpenReact**, because it is a monoid in **Mfd** and **Mfd** embeds into **OpenReact**, and thus $(-) \times \mathbb{R}$ is a monad on **OpenReact**.

Definition 12. Let **OpenErg** be the symmetric monoidal category of *open energy-driven systems*, defined as the Kleisli category of $(-) \times \mathbb{R}$ on **OpenReact**.

A 1-cell in **OpenErg** from A to B is then a state space X with a reaction $R \in \mathbf{React}(X)$ and a smooth function $\langle w, E \rangle: A \times X \rightarrow B \times \mathbb{R}$, which is precisely an open energy-driven system as in Definition 7.

The composition of $(X, R, \langle w, E \rangle): A \times X \rightarrow B \times \mathbb{R}$ and $(X', R', \langle w', E' \rangle): B \times X' \rightarrow C \times \mathbb{R}$ is the energy-driven system:

$$(X \times X', R \oplus R', \langle w \circ w', E + w^* E' \rangle): A \times (X \times X') \rightarrow C \times \mathbb{R} \quad (6)$$

where $w^* E': A \times (X \times X') \rightarrow \mathbb{R}$ is given by $(a, x, x') \mapsto E'(w(a, x), x')$ and we abuse notation by writing E for $\pi_{A, X} \circ E$. The symmetric monoidal structure is defined similarly as above.

Remark 13. There are more natural higher-category structures that we could use instead of just symmetric monoidal categories. For instance, the Para construction naturally produces a double category. In fact the generalized Para construction of [Mye23b, CM23] can even define **OpenErg** in one fell swoop by having the component $E: A \times X \rightarrow \mathbb{R}$ be part of the decoration on the states.

4 The Semantics of Open Energy-driven Systems

In order to develop semantics for open energy-driven systems, we must have some notion of an open ODE. It is the job of the current section to construct this.

¹Taking the **local 0-truncation** of a bicategory means we replace each hom-category with the set of its isomorphism classes. Notice this yields a well-defined strict 1-category since coherence isomorphisms are turned into equalities.

4.1 Open Ordinary Differential Equations

Definition 14. A smooth function $p: \bar{X} \rightarrow X$ is a *submersion* if all $(Tp)_{\bar{x}}: T_{\bar{x}}\bar{X} \rightarrow T_{p(\bar{x})}X$ are surjective.

Proposition 15. If $p: \bar{X} \rightarrow X$ is a submersion, then all pullbacks along p exist, and are computed as in the category of topological spaces.

Proof. Standard, can be found in [KMS93, Corollary I.2.19]. $\square$

Definition 16. Let $\mathbf{Subm}: \mathbf{Mfd}^{\text{op}} \rightarrow \mathbf{Cat}$ be the pseudofunctor which sends a space X to the category of submersions $\bar{X} \rightarrow X$ over it, and a smooth function $f: X \rightarrow Y$ to its action by pullback $f^*: \mathbf{Subm}(Y) \rightarrow \mathbf{Subm}(X)$.

Definition 17. Let $\mathbf{MfdLens}$ be the symmetric monoidal category of **Subm-lenses**, defined in [Spi19] as $\int \mathbf{Subm}^{\text{op}}$, meaning its objects are given by submersions $p: \bar{X} \rightarrow X$ and denoted by $\left(\frac{\bar{X}}{X}\right)$ and its 1-cells, denoted as below left, are given by pairs of dashed arrows as below right:

$$\left(\frac{f^\sharp}{f}\right): \left(\frac{\bar{X}}{X}\right) \hookrightarrow \left(\frac{\bar{Y}}{Y}\right) \quad \begin{array}{ccccc} \bar{X} & \xleftarrow{f^\sharp} & f^*\bar{Y} & \longrightarrow & \bar{Y} \\ p \downarrow & & \downarrow & \lrcorner & \downarrow q \\ X & \xlongequal{\quad} & X & \xrightarrow{\quad f \quad} & Y \end{array} \quad (7)$$

The symmetric monoidal product of this category is given by fibrewise product of bundles.

The reader might notice that the form of these **Subm-lenses** is exactly that of the pullback map on covectors induced between the cotangent bundles by any smooth map, i.e. if $f: X \rightarrow Y$ is a smooth map there is a corresponding lens

$$\begin{array}{ccccc} T^*X & \xleftarrow{T^*f} & f^*T^*Y & \longrightarrow & T^*Y \\ p \downarrow & & \downarrow & \lrcorner & \downarrow q \\ X & \xlongequal{\quad} & X & \xrightarrow{\quad f \quad} & Y \end{array}$$

In fact the assignment $f \mapsto \left(\frac{T^*f}{f}\right)$ defines a functor $T^*: \mathbf{Mfd} \rightarrow \mathbf{MfdLens}$, which will be central later.

We want to interpret an open energy-driven system on A, B with state space X as a parametric lens:

$$\left(\frac{T^*A}{A}\right) \otimes \left(\frac{T^*X}{X}\right) \hookrightarrow \left(\frac{T^*B}{B}\right)$$

where the parameter space X is decorated by the data of a reaction, seen as an open ODE $\left(\frac{TX}{X}\right) \hookrightarrow \left(\frac{T^*X}{X}\right)$. We do this with a similar Para construction as before, though to tame coherence we directly construct the (op)fibration of $\pi_{\text{ODE}}: \mathbf{ODE} \rightarrow \mathbf{MfdLens}$ by means of an isocomma construction:

Construction 18. Let $T: \mathbf{Mfd}^{\text{iso}} \rightarrow \mathbf{MfdLens}$ be the tangent bundle functor restricted to the core of $\mathbf{Mfd}$, i.e. its wide subcategory of isomorphisms. Consider the isocomma $\mathbf{ODE} := \{T \cong \mathbf{MfdLens}\}$, performed in the 2-category of symmetric monoidal categories and symmetric strong monoidal functors (which exists well-defined by e.g. [BKP89]). This is itself a symmetric monoidal category, opfibred over $\mathbf{MfdLens}$, with fibers given by the groupoids:

$$\mathbf{ODE}\left(\frac{\bar{Y}}{Y}\right) = \left\{ X \in \mathbf{Mfd}, \left(\frac{u^\sharp}{u}\right): \left(\frac{TX}{X}\right) \hookrightarrow \left(\frac{\bar{Y}}{Y}\right) \right\}, \quad (8)$$

with morphisms given by isomorphisms $\varphi : X \cong X'$ between the state spaces such that $\begin{pmatrix} T\varphi^{-1} \\ \varphi \end{pmatrix} : \begin{pmatrix} TX \\ X \end{pmatrix} \rightleftharpoons \begin{pmatrix} TX' \\ X' \end{pmatrix}$ commutes with the dynamics on X and X' . The symmetric monoidal structure is inherited from that of **MfdLens**, and relies crucially on the fact that $T : \mathbf{Mfd}^{\text{iso}} \rightarrow \mathbf{MfdLens}$ is a strong monoidal functor.

The projection $\pi_{\text{ODE}} : \mathbf{ODE} \rightarrow \mathbf{MfdLens}$ induces an action of the former on the latter, which we denote $\otimes_{\text{ODE}}$.

Definition 19. We call **OpenODE** the local 0-truncation of $\mathbf{Para}(\otimes_{\text{ODE}})$.

Thus a map $\begin{pmatrix} \bar{A} \\ A \end{pmatrix} \rightarrow \begin{pmatrix} \bar{B} \\ B \end{pmatrix}$ in **OpenODE** is a choice of parameter interface $\begin{pmatrix} \bar{P} \\ P \end{pmatrix}$ and of an open ODE over it, say $\begin{pmatrix} u^\sharp \\ u \end{pmatrix} : \begin{pmatrix} TX \\ X \end{pmatrix} \rightleftharpoons \begin{pmatrix} \bar{P} \\ P \end{pmatrix}$, and then a choice of smooth lens $\begin{pmatrix} \bar{A} \\ A \end{pmatrix} \otimes \begin{pmatrix} \bar{P} \\ P \end{pmatrix} \rightleftharpoons \begin{pmatrix} \bar{B} \\ B \end{pmatrix}$. At this stage, the open ODE on the parameter is nothing more than a decoration.

4.2 An “organized” view

In the symmetric monoidal category **OpenODE**, systems (the ODEs) and their wiring (the parametric lens they are grafted on) are kept neatly separated. This is because, in general, the systems might be very different from their wiring, but in this case they aren’t: both open ODEs and their wiring are differential lenses, so one can collapse the data of the ODE directly in the wiring, seeing it as all part of a unique process. This brings us to define **COrg**, which is a smooth space, continuous time variant of **Org**, introduced in [Spi22]; albeit we confine ourselves to defining a 1-category and not a full double category like **Org** is.

Once again, we define an action through a symmetric monoidal functor, this time given by the tangent bundle functor $T : \mathbf{Mfd}^{\text{iso}} \rightarrow \mathbf{MfdLens}$ sending $X \mapsto \begin{pmatrix} TX \\ X \end{pmatrix}$. This defines an action of $\mathbf{Mfd}^{\text{iso}}$ on $\mathbf{MfdLens}$, which we denote $\otimes_T$.

Definition 20. We call **COrg** the local 0-truncation of $\mathbf{Para}(\otimes_T)$.

The categories **OpenODE** and **COrg** have the same objects, namely submersions $\begin{pmatrix} \bar{A} \\ A \end{pmatrix}$, but a 1-cell from $\begin{pmatrix} \bar{A} \\ A \end{pmatrix}$ to $\begin{pmatrix} \bar{B} \\ B \end{pmatrix}$ in **COrg** consists of a manifold X (up to diffeomorphism) and a lens $\begin{pmatrix} \bar{A} \\ A \end{pmatrix} \otimes \begin{pmatrix} TX \\ X \end{pmatrix} \rightleftharpoons \begin{pmatrix} \bar{B} \\ B \end{pmatrix}$.

Proposition 21. *There is an identity-on-objects, symmetric monoidal functor*

$$\text{collapse} : \mathbf{OpenODE} \longrightarrow \mathbf{COrg}$$

given by

$$\begin{array}{ccc} \begin{pmatrix} TX \\ X \end{pmatrix} & & \begin{pmatrix} TX \\ X \end{pmatrix} \xleftarrow{u^\sharp} \begin{pmatrix} \bar{P} \\ P \end{pmatrix} \\ u \downarrow \uparrow u^\sharp & \mapsto & \otimes \quad \otimes \xleftarrow{f^\sharp} \begin{pmatrix} \bar{B} \\ B \end{pmatrix} \\ \begin{pmatrix} \bar{A} \\ A \end{pmatrix} \otimes \begin{pmatrix} \bar{P} \\ P \end{pmatrix} \xleftarrow{f^\sharp} \begin{pmatrix} \bar{B} \\ B \end{pmatrix} & & \begin{pmatrix} \bar{A} \\ A \end{pmatrix} \xlongequal{\quad} \begin{pmatrix} \bar{A} \\ A \end{pmatrix} \xleftarrow{f} \begin{pmatrix} \bar{B} \\ B \end{pmatrix} \end{array}$$

Proof. The fact this is functorial and symmetric monoidal boils down to the interchange law of the monoidal structure of **MfdLens**, which allows to map composition in **OpenODE** (which puts the ODEs in the parameter side by side) to composition in **COrg** (which composes sequentially the 1-cells). $\square$

In fact, this definition applies already to discrete systems, i.e. to (the horizontal 1-category of) $\mathbf{Org}$, which we denote as **Org**. Recall this 1-category is the symmetric monoidal category whose objects are polynomial functors $p \in \mathbf{Poly}$ and whose 1-cells $p \rightarrow q$ are parametric maps of polynomials (i.e. parametric dependent lenses) of the form $p \otimes Sy^S \rightarrow q$, thus where the parameter is given by something of the form Sy^S (*state systems* in the language of [NS23]), and these compose in a Para way.

Starting with **Poly**, define a functor **Coalg**: **Poly** $\rightarrow$ **Set** which sends a polynomial to its (large) set of coalgebras. Applying the Para construction to the forgetful functor $\int \mathbf{Coalg} \rightarrow \mathbf{Poly}$, we get something similar to **OpenODE**, where a 1-cell from p to q is a polynomial r , a coalgebra $\delta : S \rightarrow r(S)$, and a poly map $\left(\begin{smallmatrix} f^\# \\ f \end{smallmatrix} \right) : p \otimes r \rightarrow q$. Now, the coalgebra δ is in fact equivalent to a map $\hat{\delta} : Sy^S \rightarrow r$, and thus we can define a symmetric monoidal functor **Para**($\otimes_{\mathbf{Coalg}}$) $\rightarrow$ **Org** which collapses the pair $(\delta, \left(\begin{smallmatrix} f^\# \\ f \end{smallmatrix} \right))$ to a single map $p \otimes Sy^S \rightarrow q$.

Generally speaking, this trick works whenever the indexed set of systems **Sys** : **C** $\rightarrow$ **Set** is ‘free’ in a specific sense: its elements are the objects of the slice $T/\mathbf{C}$, there T is a functor $T : \mathbf{States} \rightarrow \mathbf{C}$ which picks out “state spaces”,² not unlike what we did in Construction 18. In that case we can always build a collapse functor **Para**($\otimes_{\mathbf{Sys}}$) $\rightarrow$ **Para**($\otimes_T$) by reproducing the construction of Proposition 21. Systems which are given by slicing under a functor play a central role in [Mye23a], where Myers shows most theories of systems can be obtained in this way.

4.3 The Cotangent Functor

The aim of this section is to show how the cotangent functor $T^* : \mathbf{Mfd} \rightarrow \mathbf{MfdLens}$ induces a functor **OpenErg** $\rightarrow$ **OpenODE**:

$$\begin{array}{ccc}
 R \in \mathbf{React}(X) & \mapsto & \left(\begin{smallmatrix} TX \\ X \end{smallmatrix} \right) \in \mathbf{ODE} \left(\begin{smallmatrix} T^*X \\ X \end{smallmatrix} \right) \\
 A \times X \xrightarrow{\langle w, E \rangle} B \times \mathbb{R} & & \begin{array}{c} \parallel \uparrow R \\ \left(\begin{smallmatrix} T^*A \\ A \end{smallmatrix} \right) \times \left(\begin{smallmatrix} T^*X \\ X \end{smallmatrix} \right) \xleftarrow[T^*w + dE]{w} \left(\begin{smallmatrix} T^*B \\ B \end{smallmatrix} \right) \end{array}
 \end{array} \tag{9}$$

The functor could be obtained in one single step using the technology of the generalized Para construction we hinted at in Remark 13 with which we can see both **OpenErg** and **OpenODE** as Para constructions and thus induce the desired functor by exhibiting one between the underlying fibred actions (see [CM23]). Since we don’t have the space to introduce this machinery, we just give a direct construction.

This would make the analogy with [Cap23] total since that is how the functor from game descriptions to parametric lenses is obtained. Notably, in this case T^* is strong monoidal thus making the semantics of open energy-driven systems truly compositional.

Theorem 22. *The assignment defined in (9) is a well-defined symmetric monoidal functor*

$$\mathbf{T}^* : \mathbf{OpenErg} \rightarrow \mathbf{OpenODE}.$$

Proof. It is easy to see it sends identities to identities. Given composable open energy-driven systems $(X, R, \langle w, E \rangle : A \times X \rightarrow B \times \mathbb{R})$ and $(X', R', \langle w', E' \rangle : B \times X' \rightarrow C \times \mathbb{R})$, their images are the composable

²Being a “state space” is an attitude, not a formal mathematical concepts: any functor into **C** suffices.

open ODEs

$$\begin{array}{ccc}
 \begin{pmatrix} TX \\ X \end{pmatrix} & & \begin{pmatrix} TX' \\ X' \end{pmatrix} \\
 \parallel \uparrow R & \circ & \parallel \uparrow R' \\
 \begin{pmatrix} T^*A \\ A \end{pmatrix} \otimes \begin{pmatrix} T^*X \\ X \end{pmatrix} & \xleftarrow[w]{T^*w + dE} & \begin{pmatrix} T^*B \\ B \end{pmatrix} & \begin{pmatrix} T^*B \\ B \end{pmatrix} \otimes \begin{pmatrix} T^*X' \\ X' \end{pmatrix} & \xleftarrow[w']{T^*w' + dE'} & \begin{pmatrix} T^*C \\ C \end{pmatrix}
 \end{array}$$

which reduces to

$$\begin{array}{ccc}
 \begin{pmatrix} T(X \times X') \\ X \times X' \end{pmatrix} \\
 \parallel \uparrow R \oplus R' \\
 \begin{pmatrix} T^*A \\ A \end{pmatrix} \otimes \begin{pmatrix} T^*(X \times X') \\ X \times X' \end{pmatrix} & \xleftarrow[w]{T^*w + dE} & \begin{pmatrix} T^*B \\ B \end{pmatrix} \otimes \begin{pmatrix} T^*X' \\ X' \end{pmatrix} & \xleftarrow[w']{T^*w' + dE'} & \begin{pmatrix} T^*C \\ C \end{pmatrix}
 \end{array}$$

Given $a \in A, x \in X, x' \in X'$, the composite backward map sends a covector $\alpha \in T^*C$ (we omit indexing of bundles for brevity) to

$$T^*w(T^*w'(\alpha) + dE'(w(a, x), x')) + dE(a, x) = T^*(w \circ w')(\alpha) + d(w^*E' + E)(a, x, x').$$

The latter expression equals the image of the composite energy-driven system $(X \times X', R \oplus R', \langle w \circ w', E + w^*E' \rangle : A \times (X \times X') \rightarrow C \times \mathbb{R})$:

$$\begin{array}{ccc}
 \begin{pmatrix} T(X \times X') \\ X \times X' \end{pmatrix} \\
 \parallel \uparrow R \oplus R' \\
 \begin{pmatrix} T^*A \\ A \end{pmatrix} \otimes \begin{pmatrix} T^*(X \times X') \\ (X \times X') \end{pmatrix} & \xleftarrow[w \circ w']{T^*(w \circ w') + d(E + w^*E')} & \begin{pmatrix} T^*C \\ C \end{pmatrix}
 \end{array}$$

Preservation of symmetric monoidal structure is trivial, as it amounts to no more than the analogue structure on T^* . $\square$

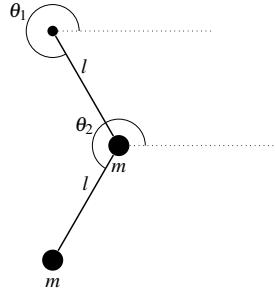
We can further collapse the description of the dynamics of the system by folding the ODE into the map itself, thus landing in **COrg**:

$$\mathbf{OpenErg} \xrightarrow{T^*} \mathbf{OpenODE} \xrightarrow{\text{collapse}} \mathbf{COrg}.$$

5 Examples

We motivated the last three sections by saying that we were going to compose the pendulum with itself to get an n -fold pendulum. We do this now.

Example 23. For intuition, observe the following diagram of the double pendulum. We will work out carefully the composition of two pendulums to create a double pendulum, and leave the iterated composition to the reader.



Recall that the single pendulum in **OpenErg** as defined in Example 6 is an endomorphism $T\mathbb{R}^2 \rightarrow T\mathbb{R}^2$, whose state space is T^*S^1 and whose reaction structure is derived from the canonical symplectic structure and written in coordinate form as $J(x) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$.

Composing this morphism with itself involves several steps, where we unwind all of the constructions that we've done so far. First of all, we have to take the monoidal product of (T^*S^1, J) with itself in $\int \mathbf{React}$. This uses the lax monoidal structure of **React** to produce a reaction $J^{(2)} = J \oplus J$ on $T^*S^1 \times T^*S^1$. In coordinates, this is the block-diagonal matrix

$$J^{(2)}(x) = \begin{bmatrix} J(x) & 0 \\ 0 & J(x) \end{bmatrix}$$

Transforming this along the isomorphism $T^*S^1 \times T^*S^1 \cong T^*(S^1 \times S^1)$, this becomes the standard

$$J^{(2)}(x) = \begin{bmatrix} 0 & I_2 \\ -I_2 & 0 \end{bmatrix}$$

This gives a new parameter space; we now compose the smooth maps $w: T\mathbb{R}^2 \times TS^1 \rightarrow T\mathbb{R}^2$ and $E: T\mathbb{R}^2 \times TS^1 \rightarrow \mathbb{R}$ with themselves to get $w^{(2)}: T\mathbb{R}^2 \times (T^*S^1 \times T^*S^1) \rightarrow T\mathbb{R}^2$ defined by

$$T\mathbb{R}^2 \times T^*S^1 \times T^*S^1 \xrightarrow{w \times 1_{T^*S^1}} T\mathbb{R}^2 \times T^*S^1 \xrightarrow{w} T\mathbb{R}^2$$

and $E^{(2)}: T\mathbb{R}^2 \times (T^*S^1 \times T^*S^1) \rightarrow \mathbb{R}$ defined by

$$T\mathbb{R}^2 \times T^*S^1 \times T^*S^1 \xrightarrow{\langle E, w \times 1_{T^*S^1} \rangle} \mathbb{R} \times T\mathbb{R}^2 \times T^*S^1 \xrightarrow{1_{\mathbb{R}} \times E} \mathbb{R} \times \mathbb{R} \xrightarrow{+} \mathbb{R}$$

In coordinates, this looks like the following. Assume that x_0, v_0 are the position and velocity of the first pivot, and $\theta_1, L_1, \theta_2, L_2$ are the natural coordinates for $T^*S^1 \times T^*S^1$. Then we define the following quantities, all of which depend on $(x_0, v_0, \theta_1, L_1, \theta_2, L_2) \in T\mathbb{R}^2 \times T^*S^1 \times T^*S^1$,

$$\begin{aligned} x_1 &= x_0 + l(\cos \theta_1, \sin \theta_1) \\ \omega_1 &= \frac{L_1}{I} \\ v_1 &= v_0 + l\omega_1(-\sin \theta_1, \cos \theta_1) \\ x_2 &= x_1 + l(\cos \theta_2, \sin \theta_2) \\ \omega_2 &= \frac{L_2}{I} \\ v_2 &= v_1 + l\omega_2(-\sin \theta_2, \cos \theta_2), \end{aligned}$$

in order to define the output $w^{(2)}$ and energy $E^{(2)}$.

$$w^{(2)} = (x_2, v_2)$$

$$E^{(2)} = \frac{1}{2}m|v_1|^2 + \frac{1}{2}m|v_2|^2 + mgh_1 + mgh_2.$$

We finally apply the functor **OpenErg** $\rightarrow$ **COrg** in order to get an open dynamical system out of this. The state variables of this dynamical system are $(\theta_1, L_1, \theta_2, L_2)$. It is a not-terribly-interesting exercise in calculus to actually compute all of the partials for $w^{(2)}$ and $E^{(2)}$; the interested reader could run the following Julia script if they wanted to see the actual expressions.

using Symbolics

```
@variables x0 y0 vx0 vy0 θ1 L1 θ2 L2
```

```
@variables l I m g
```

```
x1 = x0 + l * cos(θ1)
```

```
y1 = y0 + l * sin(θ1)
```

```
ω1 = L1 / I
```

```
vx1 = vx0 + l * ω1 * (- sin(θ1))
```

```
vy1 = vy0 + l * ω1 * cos(θ1)
```

```
x2 = x1 + l * cos(θ2)
```

```
y2 = x1 + l * sin(θ2)
```

```
ω2 = L2 / I
```

```
vx2 = vx1 + l * ω2 * (- sin(θ2))
```

```
vy2 = vy1 + l * ω2 * cos(θ2)
```

```
E = 0.5 * m * (vx1^2 + vy1^2) + 0.5 * m * (vx2^2 + vy2^2) + m * g * y1 + m * g * y2
```

```
dEdx0 = expand_derivatives(Differential(x0)(E))
```

```
dEdy0 = expand_derivatives(Differential(y0)(E))
```

```
dEdvx0 = expand_derivatives(Differential(vx0)(E))
```

```
dEdvy0 = expand_derivatives(Differential(vy0)(E))
```

```
dEdθ1 = expand_derivatives(Differential(θ1)(E))
```

```
dEdL1 = expand_derivatives(Differential(L1)(E))
```

```
dEdθ2 = expand_derivatives(Differential(θ2)(E))
```

```
dEdL2 = expand_derivatives(Differential(L2)(E))
```

```
dθ1dt = dEdL1
```

```
dL1dt = -dEdθ1
```

```
dθ2dt = dEdL2
```

```
dL2dt = -dEdθ2
```

6 Epilogue

In this work, we have shown the structure of open energy-driven system, involving a reaction and an energy functional, elegantly subsumes both Hamiltonian and gradient-based systems. Out of these, we have built a symmetric monoidal category **OpenErg** which maps to **OpenODE**, a symmetric monoidal category of open ODEs, which itself maps to **COrg**, whose 1-cells are lenses ‘evolving smoothly’ according to a given state space.

While this may seem like a lot of work/abstraction for little gain compared to “just doing physics” in the way one might in a traditional classical mechanics course, the advantage of working out this theory is to produce a “plug and play” physics system, where a library of components can be intuitively composed and all of the algebra is done by the computer.

References

- [Arn89] Vladimir Igorevich Arnol’d (1989): *Mathematical Methods of Classical Mechanics*, 2 edition. Springer New York, doi:[10.1007/978-1-4757-2063-1](https://doi.org/10.1007/978-1-4757-2063-1).
- [BKP89] Robert Blackwell, Gregory M. Kelly & A. John Power (1989): *Two-dimensional monad theory*. *Journal of pure and applied algebra* 59(1), pp. 1–41, doi:[10.1016/0022-4049\(89\)90160-6](https://doi.org/10.1016/0022-4049(89)90160-6).
- [BWY21] John C. Baez, David Weisbart & Adam M. Yassine (2021): *Open systems in classical mechanics*. *Journal of Mathematical Physics* 62(4), doi:[10.1063/5.0029885](https://doi.org/10.1063/5.0029885).
- [Cap23] Matteo Capucci (2023): *Diegetic representation of feedback in open games*. *Electronic Proceedings in Theoretical Computer Science* 380, pp. 145–158, doi:[10.4204/EPTCS.380.9](https://doi.org/10.4204/EPTCS.380.9).
- [CFM21] Marius Crainic, Rui Loja Fernandes & Ioan Mărcuț (2021): *Lectures on Poisson geometry*. 217, American Mathematical Soc., doi:[10.1090/gsm/217](https://doi.org/10.1090/gsm/217).
- [CGG⁺22] Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson & Fabio Zanasi (2022): *Categorical Foundations of Gradient-Based Learning*. In Ilya Sergey, editor: *Programming Languages and Systems*, Springer International Publishing, Cham, pp. 1–28, doi:[10.1007/978-3-030-99336-8_1](https://doi.org/10.1007/978-3-030-99336-8_1).
- [CGHR22] Matteo Capucci, Bruno Gavranović, Jules Hedges & Eigil Fjeldgren Rischel (2022): *Towards Foundations of Categorical Cybernetics*. *Electronic Proceedings in Theoretical Computer Science* 372, pp. 235–248, doi:[10.4204/eptcs.372.17](https://doi.org/10.4204/eptcs.372.17).
- [CM23] Matteo Capucci & David Jaz Myers (2023): *Constructing Triple Categories of Cybernetic Processes*, doi:[10.5281/zenodo.8215544](https://doi.org/10.5281/zenodo.8215544).
- [FST19] Brendan Fong, David Spivak & Remy Tuyeras (2019): *Backprop as Functor: A compositional perspective on supervised learning*. In: *2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, IEEE, doi:[10.1109/lics.2019.8785665](https://doi.org/10.1109/lics.2019.8785665).
- [HT12] C. Hermida & R. D. Tennent (2012): *Monoidal indeterminates and categories of possible worlds*. *Theoretical Computer Science* 430, pp. 3–22, doi:[10.1016/j.tcs.2012.01.001](https://doi.org/10.1016/j.tcs.2012.01.001).
- [KMS93] Ivan Kolar, Peter W Michor & Jan Slovak (1993): *Natural operations in differential geometry*, 1993 edition. Springer, Berlin, Germany, doi:[10.1007/978-3-662-02950-3](https://doi.org/10.1007/978-3-662-02950-3).
- [Law80] F William Lawvere (1980): *Toward the description in a smooth topos of the dynamically possible motions and deformations of a continuous body*. *Cahiers de topologie et géométrie différentielle* 21(4), pp. 377–392.
- [LBPF22] Sophie Libkind, Andrew Baas, Evan Patterson & James Fairbanks (2022): *Operadic Modeling of Dynamical Systems: Mathematics and Computation*. *Electronic Proceedings in Theoretical Computer Science* 372, pp. 192–206, doi:[10.4204/eptcs.372.14](https://doi.org/10.4204/eptcs.372.14).

- [LLL24] Markus Lohmayer, Owen Lynch & Sigrid Leyendecker (2024): *Exergetic Port-Hamiltonian Systems Modeling Language*, doi:[10.48550/ARXIV.2402.17640](https://doi.org/10.48550/ARXIV.2402.17640).
- [Lyn22] Owen Lynch (2022): *Relational Composition of Physical Systems: A Categorical Approach*. M.sc. thesis. Available at <https://studenttheses.uu.nl/handle/20.500.12932/41725>.
- [MGA⁺21] Yingbo Ma, Shashi Gowda, Ranjan Anantharaman, Chris Laughman, Viral Shah & Chris Rackauckas (2021): *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. arXiv:[2103.05244](https://arxiv.org/abs/2103.05244).
- [MV20] Joe Moeller & Christina Vasilakopoulou (2020): *Monoidal Grothendieck Construction*. *Theory and Applications of Categories* 35(31), pp. 1159–1207.
- [Mye23a] David Jaz Myers (2023): *Categorical Systems Theory*. Available at <http://davidjaz.com/Papers/DynamicalBook.pdf>.
- [Mye23b] David Jaz Myers (2023): *The Para construction as a Wreath Product*. Available at https://davidjaz.com/Talks/CT2023_Para_as_Wreath_Product.pdf.
- [NS23] Nelson Niu & David I. Spivak (2023): *Polynomial Functors: A Mathematical Theory of Interaction*, doi:[10.48550/arXiv.2312.00990](https://doi.org/10.48550/arXiv.2312.00990). ArXiv:2312.00990 [math].
- [Sou06] Jean-Marie Souriau (2006): *Groupes différentiels*. In: *Differential Geometrical Methods in Mathematical Physics: Proceedings of the Conferences Held at Aix-en-Provence, September 3–7, 1979 and Salamanca, September 10–14, 1979*, Springer, pp. 91–128, doi:[10.1007/BFb0089728](https://doi.org/10.1007/BFb0089728).
- [Spi19] David I. Spivak (2019): *Generalized Lens Categories via Functors $C^{\text{op}} \rightarrow \text{Cat}$* , doi:[10.48550/ARXIV.1908.02202](https://doi.org/10.48550/ARXIV.1908.02202).
- [Spi22] David I. Spivak (2022): *Learners’ Languages*. *Electronic Proceedings in Theoretical Computer Science* 372, pp. 14–28, doi:[10.4204/eptcs.372.2](https://doi.org/10.4204/eptcs.372.2).
- [SS23] Brandon T. Shapiro & David I. Spivak (2023): *Dynamic Operads, Dynamic Categories: From Deep Learning to Prediction Markets*. *Electronic Proceedings in Theoretical Computer Science* 380, pp. 183–202, doi:[10.4204/eptcs.380.11](https://doi.org/10.4204/eptcs.380.11).