Monoidal Context Theory

Mario Román

Tallinn University of Technology, PhD Thesis,
Defended on the 16th November 2023, Tallinn
Supervisor: Paweł Sobociński
Opponents: Guy McCusker and Paul-André Melliès

Abstract. We universally characterize the produoidal category of monoidal lenses over a monoidal category. In the same way that each category induces a cofree promonoidal category of spliced arrows, each monoidal category induces a cofree produoidal category of monoidal spliced arrows; monoidal lenses are the free normalization of the cofree produoidal category of monoidal spliced arrows.

We apply the characterization of symmetric monoidal lenses to the analysis of multi-party message-passing protocols. We introduce a minimalistic axiomatization of message passing – message theories – and we construct combinatorially the free message theory over a set. Symmetric monoidal lenses are the derivations of the free message theory over a symmetric monoidal category.

Monoidiliste Kontekstide Teooria

Kokkuvõte. Karakteriseerime monoidiliste läätsede produoidilise kategooria universaalomaduse abil. Nii nagu iga kategooria indutseerib pleissitud noolte kovaba promonoidilise kategooria, indutseerib monoidiline kategooria monoidiliste pleissnoolte kovaba produoidilise kategooria; monoidilised läätsed on monoidiliste pleissnoolte kovaba produoidilise kategooria vaba normalisatsioon.

Kasutame sümmeetriliste monoidiliste läätsede karakterisatsiooni mitme osapoole sõnumiedastusprotokollide analüüsimiseks. Toome sisse sõnumiedastuse minimalistliku aksiomatisatsiooni – sõnumiteooriaid – ja konstrueerime vaba sõnumiteooria etteantud hulgal. Sümmeetrilised monoidilised läätsed on sümmeetrilise monoidilise kategooria vaba sõnumiteooria tuletised.

Acknowledgements. I would like to thank my PhD advisor, Pawel Sobocinski. Pawel has an exceptional ability to separate the scientifically promising ideas from the noise; he gave me the support, encouragement and freedom to pursue the research on this thesis. Pawel always said he wanted to replicate in Tallinn the atmosphere of Bob Walters' group in Sydney and I am particularly thankful for the result. I am also very grateful to Nicoletta Sabadini, for her advice and for sharing her encyclopedic knowledge of both automata and the history of Como.

Most ideas were cultivated at group meetings, and I want to thank Ed, Chad, Clémence, Nathan, Diana, Fosco, Elina, Amar, Cole, Philipp, Ekaterina, Niccolò, Michele, Andrea, and the rest of the *Tarkvarateaduse Instituut* for all the math and time we shared. I am very grateful to Niels, for his contagious enthusiasm and much useful feedback on this thesis. Special thanks go to Matt for his attention to detail and mathematical elegance, great discussions and ideas, and equally great book recommendations.

I learned and enjoyed a lot on short but productive visits to Pisa, Como, Oxford and Paris, and I want to thank Filippo, Alessandro, Vladimir, Louis and Davidad for many insightful discussions during this thesis. I had the privilege of having Giovanni, James, and Dylan as coauthors and I learned a lot from each one of them.

I thank the constant support of my parents, my brother Víctor, and my friends; I especially thank David and Esperanza for finding the best cafés in Granada. I thank Anna, Paolo, Enrico and Andrea for the time at the lake. Finally, I thank Elena: for all the fun we had writing each joint paper, and for all the happiness, math and drawings of these four years.

Contents

Preface

Understanding and correctly designing intelligent and explainable systems could be both, if we get it right, one of the most beneficial human advancements; and, if we get it wrong, an existential risk for humanity [Ord20]. Humanity's need for languages and formalisms for trustworthy complex systems is now an urge.

Mathematics may possibly be the only right tool for this; but mathematics has not always been concerned with complex and interconnected systems. John von Neumann, talking about the intelligent and complex system that is the human brain, famously noted that

> the outward forms of our mathematics are not absolutely relevant from the point of view of evaluating what the mathematical or logical language truly used by the central nervous system is. However, the above remarks about reliability and logical and arithmetical depth prove that whatever the system is, it cannot fail to differ considerably from what we consciously and explicitly consider as mathematics.
>
> – John Von Neumann, *The Computer and The Brain* [vN58].

Meanwhile, when we try to describe big interconnected networks with linear algebra, geometry and calculus, even with all of our achievements, we seem to miss the point: things get extremely complicated, computationally intractable, humanly unimaginable; and we declare our defeat, we resort to vague analogies, and we ask an impenetrable pile of linear algebra to be our oracle.

This does not need to be our strategy: mathematics and computer science do not advance with bigger computations; they advance with new conceptual understanding. The past century saw the rise of conceptual mathematics and theoretical computer science – the kind of mathematics that took seriously the most elementary notions and cultivated them to tame complex abstractions and systems [LS09, PC07, Gro85]. Slowly but surely, the development of the conceptual theory of categories has brought us to a point where we can forget about comforting but vague analogies and start talking about complex systems formally and scientifically.

This thesis is part of the ongoing effort to find better languages and reasoning tools for science, epistemology, causality and probability: both intuitive graphical syntaxes for humans to reason with, and formal languages for computers, linked by a trusted and transparent mathematical formalism.

Introduction

Processes and Diagrams. Processes come intuitively to us; descriptions of processes arose independently all across science and engineering, in the form of diagrams, flowcharts or prose. We reason with them and we depict them all the time, but that does not mean that we always know how to interpret them: many diagrams in computer science and elsewhere do not have clear formal semantics, so we relegate them to serving merely as sources of intuition and inspiration.

> The notation has been found very useful in practice as it greatly simplifies the appearance of complicated tensor or spinor equations, the various interrelations expressed being discernable at a glance. Unfortunately the notation seems to be of value mainly for private calculations because it cannot be printed in the normal way. – Penrose and Rindler, *Spinors and Spacetime* [PR84]

Diagrams deserve better: we can lift diagrams from mere intuitions to mathematical structures; we can defend the legitimate and exceptional conceptual mathematics we now have to talk about processes and diagrams. This thesis follows the framework of symmetric monoidal categories. Processes that pass resources around and that compose sequentially and in parallel form symmetric monoidal categories; diagrams that depict these processes are no less than a sound and complete formal syntax for symmetric monoidal categories (e.g. Figure 1).

We will develop formal syntaxes for the compositional description of process, in particular for – but not restricted to – probabilistic, effectful and non-classical processes. We make use of category theory as a foundational tool: category theory allows us to characterize a syntactic construction as the one generating a universal semantics object and, at the same time, it provides a robust classification framework for mathematical structures.

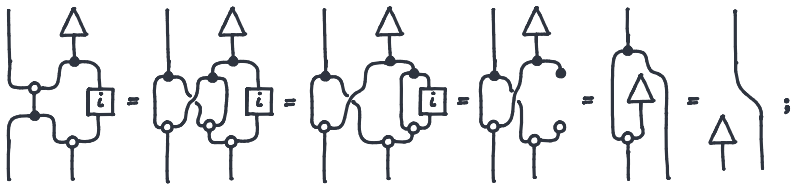


FIGURE 1. String-diagrammatic correctness proof for the One-time pad protocol (Proposition 3.5.10, [BK22]).

Algebra and Duoidal Algebra. The main technical idea of this thesis is natural: in the same way that the analysis of classical algebraic theories required the development of multicategories – and more precisely, of cartesian multicategories and Lawvere theories – the analysis of process theories, which are themselves two-dimensional algebraic theories, requires the development of monoidal multicategories and duoidal categories.

Multicategories, or colored operads, are mathematical structures that describe algebraic theories. In 1963, Lawvere introduced a categorical approach to universal algebra [Law63]: a theory can be captured by the cartesian multicategory containing all of its derived operations, and this notion is invariant to the specific primitive operations we choose to present the theory. This idea opens the field of functorial semantics: theories are categories, models are functors, and homomorphisms are natural transformations. More importantly, Lawvere's thesis gives a

robust account of classical algebra that can be modified to suit our needs: the same framework can be employed for deductive systems [Lam69], higher-order algebra [Lam86], relational algebra [BPS17], or partial algebra [DLLNS21].

How does it apply to process theories? Monoidal categories and multicategories are not structured enough for the task of describing 2-dimensional structures themselves: we need duoidal categories and produoidal categories [Str12]. Intermediate algebraic expressions with variables are not complete expressions; they are only *contexts* into which we can plug values, and context is of central importance in computer science: we model not only processes but also the environment in which they act. While the algebra of 1-dimensional context is commonplace in applications like parsing [MZ22], the same concept was missing for 2-dimensional syntaxes, which are still less frequent in computer science [UVZ18, ES22].

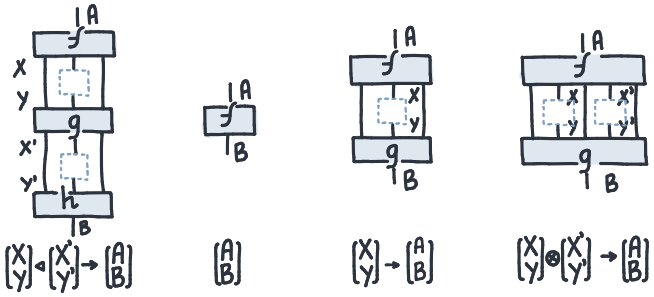


FIGURE 2. A depiction of monoidal lenses, or incomplete processes.

Duoidal categories are well-known and there is a reasonable body of literature primarily concerned with applications in pure algebra and algebraic topology [AM10, Str12]; but the usage of duoidal categories to study processes is less frequent: two notable examples are the treatment of commutativity in the work of Garner and López Franco [GF16], and the study of "compositional dependencies" in the recent work of Spivak and Shapiro [SS22]. In this text, duoidal categories and monoidal multicategories allow us to postulate axioms for modularity and message passing; these axioms apply to any symmetric monoidal category, or any process theory.

Fundamental Structures for Message Passing. This main idea has an immediate consequence that we explore in the second part of this thesis: we can now develop an algebra for incomplete processes and their communication. While *concurrent* software has been intensively studied since the early 60s, the theoretical research landscape remains quite fragmented: we do not have a satisfactory understanding of the underlying mathematical principles of concurrency, and the proliferation of models has not helped us understand how they relate. Indeed, Abramsky [Abr05] argued in 2006 that we simply do not know what the fundamental structures of concurrency are.

A way to identify such principles and arrive at more canonical models is to look for logical or universal properties. An example of the former is the discovery of and work on Curry-Howard style connections between calculi for concurrency and fragments of linear logic, which led to the development of session types [Hon93, Dd09]. We take the latter route: departing from monoidal categories and their theory of context, we universally characterize a minimalistic axiomatization of message passing in process theories.

Concurrent message passing assumes two principles: interleaving and polarization. Polarization is a categorical technique to construct dualities; and in message passing, it constructs the duality between sending and receiving [CS07, Nes21, Mel21]. Interleaving is well-known in concurrency, and it models the ability of multiple processes to advance in parallel by mixing their global effects: imagine multiple processes determined by a sequence of statements; their concurrent execution may shuffle these statements in any possible order – the only requirement is to preserve the relative order of statements within any single process. We will not only propose a minimalistic axiomatization of message passing from these two principles, but we will also characterize the universal structures for message passing on a process theory.

Briefly, we assume polarized types, $X^\bullet$ and X°, that correspond to *sending* and *receiving*; and ordered lists of types describe sessions. Our axioms ask that *(i)* a sending port can be linked to a receiving port; *(ii)* echoing allows us to receive and then send; *(iii)* sequences of actions can be interleaved by a shuffling τ; and *(iv)* there exists a no-operation that does nothing.

$$\frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\ (\text{COM}) \qquad \frac{}{X^\circ, X^\bullet}\ (\text{SPW}) \qquad \frac{\Gamma \qquad \Delta}{\tau(\Gamma, \Delta)}\ (\text{SHF}_\tau) \qquad \frac{}{()}\ (\text{NOP})$$

FIGURE 3. Type-theoretic presentation of a message theory.

This is a naive logic of message passing, but its strength is that it can be characterized mathematically using duoidal categories and, more concretely, physical monoidal multicategories, which we introduce. This paves the way to an adjunction that characterizes the free message theory on top of any process theory. The idea is simple but powerful: in order to construct message theories, we need to add global effects for *sending* and *receiving* to our process theories [OY16]; Theorem 4.5.9 notices that the diagrams for resulting effectful process theories can be wired precisely in the ways that the minimalistic logic of message passing prescribes.

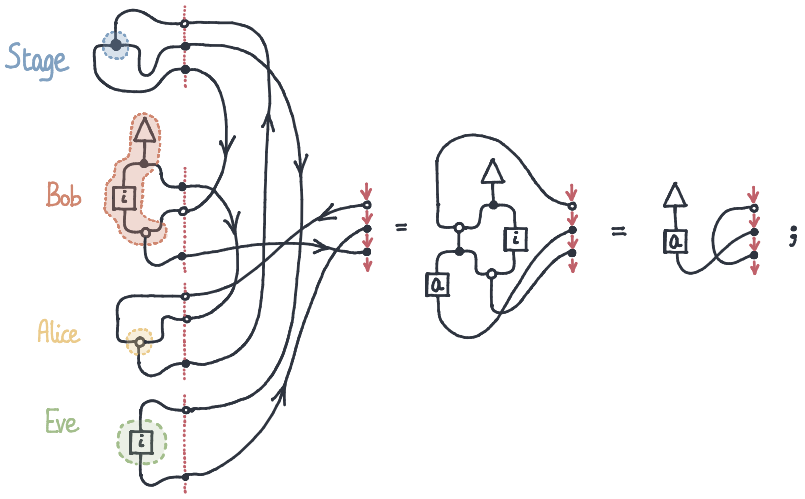


FIGURE 4. One-time pad protocol, split in four actors, mixed with a shuffle.

This means that the only addition to our process are two global effects (sending and receiving), that we depict using special red wires in the string diagrams. Each

party in a session will have one of these red wires, and the logic of message passing allows us to combine them together. For instance, if the one-time pad protocol consists of a party (say, Alice) sending a message to another party (say, Bob), with an attacker (say, Eve), sharing a Stage that only allows broadcasting of messages; then these are four parties that connect together (Figure 4).

Global Effects. It remains then to explain the idea of global effects. Most imperative programming languages assume that there exist a *global state* that the program affects. Full parallelism is not possible when two programs need to change this global state in a specific order: they could run into *race conditions* [Huf54].

However, mathematical theories of processes often assume no global state; processes do not interact with each other except when it is explicit that they do. This property is called *purity* in some functional programming languages [HJW+92] and that makes it easier to reason with them. The problem is that even pure functional programming languages need some techniques to change global state, and mathematical structures like *monads* [Mog91] or *arrows* [Hug00] achieve precisely this – they take a pure theory and endow it with global effects.

Effects, monads and arrows create premonoidal categories [Pow02, HJ06]. These are not monoidal categories, but Alan Jeffrey [Jef97a] still introduced a string diagrammatic calculus for them: it is similar to the string diagrammatic calculus of monoidal categories, but it adds a red wire to control effects. This thesis proves that the extra red wire ensures a sound and complete graphical calculus for premonoidal categories.

Monoidal Context Theory. All these ideas align to produce a theory of contexts, or incomplete processes, in monoidal categories. Each monoidal category can generate a premonoidal category with the global effects of sending and receiving. The string diagrams of this new premonoidal category can be combined using the logic of message theories, and in fact, they form the free message theory on top of the original process theory: we can use them to reason and decompose multi-party processes in arbitrary process theories.

Overview

Chapter 1: Process Theories. Chapter 1 is an introduction to monoidal categories and their string diagrammatic syntax. Section 1.1 defines strict monoidal categories in terms of process theories and introduces their string diagrams. Section 1.4 defines their *symmetric* counterpart and their type theory in terms of do-notation, while Sections 1.2 and 1.3 extend string diagrams to non-strict monoidal categories and bicategories, variants that we will employ later.

Section 1.6 is an introduction to premonoidal categories and effectful categories. Section 1.7 gives their string diagrammatic calculus and proves its soundness and completeness. Finally, Section 1.5 studies linearity, copying and discarding in terms of monoidal categories. This concludes a basic treatment of processes in terms of monoidal categories.

Chapter 2: Context Theory. Chapter 2 introduces profunctors, in Section 2.1, and multicategories, in Section 2.2, as the mathematical tools to analyze decomposition. Profunctors provide a canonical equivalence relation, dinaturality, that we use whenever we study decomposition; in fact, it brings us to consider malleable multicategories in Section 2.3. Section 2.4 presents the splice-contour adjunction between a category and its malleable multicategory of incomplete terms, or contexts.

Chapter 3: Monoidal Context Theory. Chapter 3 brings context theory to the monoidal setting. Section 3.1 and Section 3.2 introduce duoidal categories and normal duoidal categories. The duoidal counterpart of malleable multicategories are produoidal categories and we introduce their splice-contour adjunction in Section 3.3. The idempotent normalization monad of produoidal categories is constructed in Section 3.4, and it is used in Section 3.5 to normalize monoidal spliced arrows and obtain a universal characterization of monoidal lenses.

Chapter 4: Monoidal Message Passing. Chapter 4 starts defining message theories in Section 4.1. Section 4.2 studies its categorical semantics in terms of physical monoidal multicategories. Section 4.3 introduces polarization and opens the way for Section 4.4 to define polar shuffles and prove that they form a free polarized monoidal multicategory. Section 4.5 constructs an adjunction between process theories and message theories.

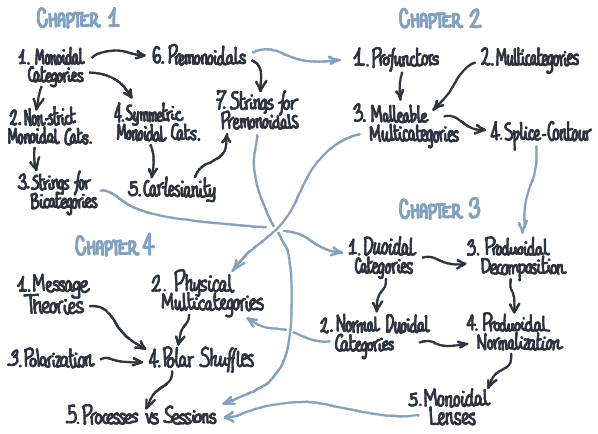


FIGURE 5. Chapter dependencies.

Contributions

The main results of this thesis are Theorem 3.5.3 and Theorem 4.5.9. They universally characterize, in two different ways, the produoidal structure of incomplete diagrams: the former is used for a theory of monoidal context, the latter is used for message passing.

The definition of *message theory* (Definitions 4.1.1 and 4.1.2 and proposition 4.1.4) is novel. There does not seem to be literature specifically on physical monoidal multicategories (Definition 4.2.4) nor on the observation that shuffles form the free one (Proposition 4.2.9) – even when, admittedly, these are all variations on the idea of physical duoidal categories and an old result by Grabowski [Gra81]. We give a different presentation of polarization in monoidal categories (Proposition 4.3.4), we discuss the problems of polarization in monoidal categories (Proposition 4.3.6) and we propose a solution describing polarization in physical monoidal multicategories (Definition 4.3.7). The definitions of polar shuffle (Definitions 4.4.1 and 4.4.2) and their physical monoidal multicategory (Theorem 4.4.7) are new contributions, as it is its proposed characterization as a free polarized physical monoidal multicategory (Theorem 4.4.11). Our main contribution is the final adjunction between sessions and processes (Theorem 4.5.9).

Duoidal categories are well-known, but we write down some observations about coherence in Proposition 3.1.6 and we contribute the definition of the physical tensor (Definition 3.2.14). Our main contribution is not only the monoidal splice-contour adjunction (Theorem 3.3.10); the adjunctions between produoidal categories and normal produoidal categories, and between symmetric produoidal categories and *physical* produoidal categories, with the construction of an idempotent monad Theorems 3.4.6 and 3.4.10, are contributions to pure category theory. Theorem 3.5.3 consitutes the first universal characterization of the whole produoidal category of lenses.

Even when do-notation is well-known, a categorical treatment like the one in Theorem 1.4.21 seemed to be missing from the literature; it is based in an exposition of string diagrams that is unusual in that it takes adjunctions as the main construction (Theorem 1.2.5). The string diagrams for premonoidal categories and effectful categories are a new formalization (Theorem 1.7.8) that is detailed in other papers by this author [Rom22]. We propose a new way of seeing coend calculus (Section 2.1.4) that is used briefly in this thesis but that is more extensively explained in other papers by this author [Rom20b]. The only contribution that we claim while translating the splice-contour adjunction to promonoidal categories is realizing their characterization as malleable multicategories (Proposition 2.3.10), which is admittedly a new spin on the usual characterization as closed multicategories.

Literature. The following is the list of publications authored or coauthored during the preparation of this thesis. As is customary in mathematics, we list authors in alphabetical order.

(1) Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregiàn, Bartosz Milewski, Emily Pillmore, and Mario Román. *Profunctor optics, a categorical update.* Accepted at *Compositionality*, preprint abs/2001.07488, 2020, [CEG+20].

(2) Mario Román. Open diagrams via coend calculus. *Applied Category Theory 2020. Electronic Proceedings in Theoretical Computer Science*, 333:65–78, Feb 2021, [Rom20b].

(3) Guillaume Boisseau, Chad Nester, and Mario Román. Cornering optics. In *Applied Category Theory 2022*, Preprint abs/2205.00842, 2022, [BNR22].
(4) Mario Román. Promonads and string diagrams for effectful categories. In Jade Master and Martha Lewis, editors, *Proceedings Fifth International Conference on Applied Category Theory, ACT 2022, Glasgow, United Kingdom, 18-22 July 2022*, volume 380 of *EPTCS*, pages 344–361, 2022, [Rom22].
(5) Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Pawel Sobocinski. A canonical algebra of open transition systems. In Gwen Salaün and Anton Wijs, editors, *Formal Aspects of Component Software - 17th International Conference, FACS 2021, Virtual Event, October 28-29, 2021, Proceedings*, volume 13077 of *Lecture Notes in Computer Science*, pages 63–81. Springer, 2021, [LGR+21].
(6) Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Pawel Sobocinski. Span(graph): a canonical feedback algebra of open transition systems. *Softw. Syst. Model.*, 22(2):495–520, 2023 [LGR+23].
(7) James Hefford and Mario Román. Optics for premonoidal categories. *Applied Category Theory 2023*, abs/2305.02906, 2023 [HR23].
(8) Elena Di Lavore, Giovanni de Felice, and Mario Román. Monoidal streams for dataflow programming. In Proceedings of the *37th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '22*, New York, NY, USA, 2022. Association for Computing Machinery. Kleene Award to the best student paper.
(9) Dylan Braithwaite and Mario Román. Collages of string diagrams. *Applied Category Theory 2023, preprint arXiv:2305.02675*, 2023 [BR23].
(10) Elena Di Lavore and Mario Román. Evidential decision theory via partial Markov categories. In *Logic In Computer Science (LICS'23)*, pages 1–14, 2023 [LR23].
(11) Matt Earnshaw, James Hefford, and Mario Román. The Produoidal Algebra of Process Decomposition, 2023. *In Peer-Review*, [EHR23].

The Produoidal Algebra of Process Decomposition is the main unpublished work (currently in peer-review) that guides the writing of the main chapter of this thesis. It develops the universal characterization of monoidal lenses and forms the basis of Chapter 3 and Chapter 4. *Promonads and String Diagrams for Effectful Categories*, adapted, was used as the basis of Section 1.6 and Section 1.7.

CHAPTER 1

Monoidal Process Theory

Monoidal Process Theory

This chapter gives an overview of monoidal categories, their variants and their syntaxes. Monoidal categories are our framework of choice for *process theories*: we claim that the minimalistic axioms of monoidal categories capture what a process theory is and we assume them for the rest of the thesis.

Section 1.1 recalls monoidal categories and their string diagrams. Section 1.2 shows that the same axioms and syntax apply to *non-strict* monoidal categories and Section 1.3 extends them to bicategories, which we will briefly use later. Section 1.4 presents our definitive notion of process theory: symmetric monoidal categories. Symmetric monoidal categories have two syntaxes that are not commonly presented together: a string diagrammatic syntax in terms of hypergraphs and a term theoretic syntax – Hughes' *do-notation* [Hug00]. We argue that these two syntaxes further justify symmetric monoidal categories as a natural setting for processes.

There is a final concept that has been traditionally left out of monoidal categories: computational effects. We argue in Sections 1.6 and 1.7 that, far from being a problem that requires an extension of monoidal categories, as usually assumed, computational effects can still use the same diagrammatic syntax of string diagrams. This will be crucial for the next chapters in *message passing*: messages will constitute a computational effect, but our results in this chapter allow us to model them without having to leave the syntax of monoidal categories.

1.1. Monoidal Categories

1.1.1. Strict Monoidal Categories. Monoidal categories are an algebra of processes, with minimal axioms. The definition of monoidal category – and this thesis – follow a particular tradition of conceptual mathematics: *category theory*. Category theory aims to extract mathematical structures in an abstract and general form. As one such structure, monoidal categories are permissive: process theories like quantum maps and Markov kernels form monoidal categories [AC09, HV19, Fri20, CJ19]; and even relations among sets or the homomorphisms of modules over a ring form monoidal categories [BSS18, Alu21]. We start by reinterpreting MacLane's axioms for a monoidal category [ML71] in terms of processes.

Definition 1.1.1. A **strict monoidal category** $\mathbb{C}$ consists of a monoid of *objects*, or resources, $(\mathbb{C}_{obj}, \otimes, I)$, and a collection of *morphisms*, or processes, $\mathbb{C}(X;Y)$, indexed by an input $X \in \mathbb{C}_{obj}$ and an output $Y \in \mathbb{C}_{obj}$. A *strict monoidal category* is endowed with operations for the sequential and parallel composition of processes, respectively

$$(\fatsemi)\colon \mathbb{C}(X;Y) \times \mathbb{C}(Y;Z) \to \mathbb{C}(X;Z),$$
$$(\otimes)\colon \mathbb{C}(X;Y) \times \mathbb{C}(X';Y') \to \mathbb{C}(X \otimes X';Y \otimes Y'),$$

and a family of *identity* morphisms, $\mathrm{id}_X \in \mathbb{C}(X;X)$. Strict monoidal categories must satisfy the following axioms.

(1) Sequencing is unital, $f \fatsemi \mathrm{id}_Y = f$ and $\mathrm{id}_X \fatsemi f = f$.
(2) Sequencing is associative, $f \fatsemi (g \fatsemi h) = (f \fatsemi g) \fatsemi h$.
(3) Tensoring is unital, $f \otimes \mathrm{id}_I = f$ and $\mathrm{id}_I \otimes f = f$.
(4) Tensoring is associative, $f \otimes (g \otimes h) = (f \otimes g) \otimes h$.
(5) Tensoring and identities interchange, $\mathrm{id}_A \otimes \mathrm{id}_B = \mathrm{id}_{A \otimes B}$.
(6) Tensoring and sequencing interchange,

$$(f \fatsemi g) \otimes (f' \fatsemi g') = (f \otimes f') \fatsemi (g \otimes g').$$

Remark 1.1.2 (Process theories)**.** Objects are also known as *types* or *resources* [CFS16]. If X and Y are both resources, it is reasonable to assume their joint occurrence is also a resource, $X \otimes Y$; this joining operation, called *tensor* $(\otimes)$, must be unital with the empty resource I. Morphisms represent *transformations* or *processes*. If we have a process transforming X into Y and a process transforming Y into Z, we can *sequence* them $(\fatsemi)$ and create a process that transforms X into Z. The process that does nothing, the identity (id), is neutral for sequential composition. Similarly, transforming X into Y and transforming X' into Y' gives a way of transforming the joint object $X \otimes X'$ into $Y \otimes Y'$. Whenever we accept these basic constructions and axioms, we end up with strict monoidal categories.

Once we have accepted these basic axioms, the next sections develop a syntax for monoidal categories: *string diagrams*. String diagrams are an intuitive syntax for process that is sound and complete for the previous axioms.

1.1.2. Some Words on Syntax. What makes a mathematical syntax practical? Different syntaxes highlight different aspects of a proof, and we consider better those that make the more bureaucratic steps invisible. Syntaxes are an explicit construction of the free mathematical object with some algebraic structure; what makes them efficient is *how* we construct them.

For instance, how to prove that, in a group, the inverse of a multiplication is the reversed multiplication of the inverses? Usually, we simply observe that

$$(x \cdot y) \cdot (y^{-1} \cdot x^{-1}) = \not{x} \cdot \cancel{y \cdot y^{-1}} \cdot \not{x}^{-1} = e;$$

that is, a simple computation checks that each letter is cancelled by its inverse. But we could be more bureaucratic and argue that the correct proof is, actually,

$$\begin{aligned}(x \cdot y) \cdot (y^{-1} \cdot x^{-1}) &\overset{(i)}{=} x \cdot (y \cdot (y^{-1} \cdot x^{-1})) \\ &\overset{(ii)}{=} x \cdot ((y \cdot y^{-1}) \cdot x^{-1}) \\ &\overset{(iii)}{=} x \cdot (e \cdot x^{-1}) \\ &\overset{(iv)}{=} x \cdot x^{-1} \\ &\overset{(v)}{=} e.\end{aligned}$$

This proof uses associativity (*i, ii*), the definition of inverse (*iii, v*), and unitality (*iv*). What makes these two proofs different? We can argue that, implicitly, they are using different syntaxes, constructed in different ways [Shu16].

The bureaucratic syntax implicitly assumes the tautological construction of a free group. The free group on a set is generated by the elements of the set, the binary multiplication ($\cdot$), the unit (e), and the inverse unary operator ($^{-1}$); then, it is quotiented by associativity, unitality, and the inverse axioms. Tautological constructions only allow bureaucratic proofs – but we can do better.

How does one construct free objects non-tautologically? The usual strategy is to first show that some combinatorial structure possesses the desired algebraic structure (say, it forms a group with some selected elements). This combinatorial structure will be as simple as possible, will relegate most steps to computation, and will use minimal quotienting. The result that makes this recipe work is freeness: the fact that it defines an adjunction (say, there exists a unique map to any group with some elements).

More concretely, in our example, we know of a better classical construction of the free group: reduced words. *Reduced words* are lists containing some generators and their inverses. The only condition is that they cannot contain ocurrences of a generator followed by its inverse: they get automatically cancelled out.

Definition 1.1.3. Given a set A, the *reduced words* over it, $\mathsf{Word}(A)$, are lists of polarized elements of A – that is, a or a^{-1} for each $a \in A$ – not containing the substrings aa^{-1} or $a^{-1}a$ for any element $a \in A$.

Definition 1.1.4. The multiplication of two reduced words is inductively defined: if the first word is empty, then the multiplication is defined to be the second, $e \cdot w_2 = w_2$; however, if the first word consists of a letter and a word, aw_1 or $a^{-1}w_1$, then we consider two cases: we first compute $w_1 \cdot w_2$ by induction; if this word starts by the inverse of the first letter, $a^{-1} \cdot w'$ or $a \cdot w'$, then they both reduce and the multiplication is $w' \cdot w_2$, otherwise, we just append the first letter, $a(w_1 \cdot w_2)$.

Remark 1.1.5. It is non-trivial to prove that this multiplication is associative: the effort we put in here is the ease we get in return every time we use the syntax. We spare the reader this proof and we focus only on showcasing the syntax.

Proposition 1.1.6. *The inverse of a multiplication is the reversed multiplication of the inverses.*

Proof. Reduced words form a group, in fact, the free group over some generators. In the group of reduced words, $(xy) \cdot (y^{-1}x^{-1}) = e$ holds by definition. Because of freeness, there is a unique group homomorphism mapping this equality to any two elements of any other group. □

The core of this argument has been to construct, combinatorially, a left adjoint $\mathsf{Words} : \mathbf{Set} \to \mathbf{Group}$ to the forgetful functor $\mathsf{Forget} : \mathbf{Group} \to \mathbf{Set}$. This thesis

will use adjoints as a more compositional way to discuss syntax. Let us start with the first of these syntaxes: string diagrams for monoidal categories.

1.1.3. String Diagrams of Strict Monoidal Categories. Monoidal categories have a sound and complete syntax in terms of string diagrams [JS91], which is the one we will use during this text. We may prefer the classical axioms of monoidal categories when proving that some category is indeed monoidal, but proving equalities in a monoidal category is easier using deformations of string diagrams – we will not need to remember the formulas. Accepting string diagrams and deformations as a criterion for equality is equivalent to accepting the axioms of strict monoidal categories: whenever we accept one, we accept the other.

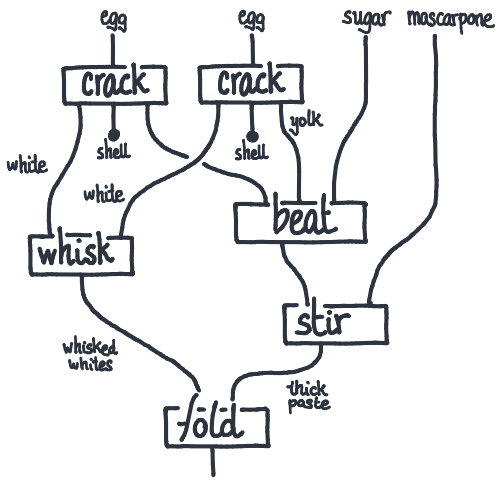


FIGURE 1. Process of preparing a Crema di Mascarpone, adapted from Sobocinski.

A first example of this syntax describing a process is in Figure 1 [Sob13]. String diagrams construct an adjunction between a category of polygraphs and a category of strict monoidal categories.

Definition 1.1.7. A *polygraph* $\mathcal{G}$ (analogue of a *multigraph* [Shu16]) is given by a set of objects, $\mathcal{G}_{obj}$, and a set of arrows $\mathcal{G}(A_0, \ldots, A_n; B_0, \ldots, B_m)$ for any two sequences of objects $A_0, \ldots, A_n$ and $B_0, \ldots, B_m$. A morphism of polygraphs $f \colon \mathcal{G} \to \mathcal{H}$ is a function between their object sets, $f_{\mathrm{o}} \colon \mathcal{G}_{obj} \to \mathcal{H}_{obj}$, and a family of functions between their corresponding morphism sets for any two sequences of objects

$$f \colon \mathcal{G}(A_0, \ldots, A_n; B_0, \ldots, B_m) \to \mathcal{H}(f_{\mathrm{o}}A_0, \ldots, f_{\mathrm{o}}A_n; f_{\mathrm{o}}B_0, \ldots, f_{\mathrm{o}}B_m).$$

Polygraphs with polygraph homomorphisms form a category, **PolyGraph**.

Definition 1.1.8. A strict *monoidal functor*, $F \colon \mathbb{C} \to \mathbb{D}$, is a monoid homomorphism between their object sets, $F_{obj} \colon \mathbb{C}_{obj} \to \mathbb{D}_{obj}$, and an assignment taking any morphism $f \in \mathbb{C}(X; Y)$ to a morphism $F(f) \in \mathbb{D}(FX; FY)$. A functor must preserve sequential composition, $F(f \fatsemi g) = F(f) \fatsemi F(g)$; parallel composition, $F(f \otimes g) = F(f) \otimes F(g)$; and identities, $F(\mathrm{id}) = \mathrm{id}$. Strict monoidal categories with strict monoidal functors form a category, $\mathbf{MonCat}_{\mathsf{Str}}$.

Definition 1.1.9. A *string diagram* over a *polygraph* $\mathcal{G}$ (or *progressive plane graph* in the work of Joyal and Street [JS91, Definition 1.1]) is a graph Γ embedded in the squared interval such that

(1) the boundary of the graph touches only the top and the bottom of the square, $\delta\Gamma \subseteq \{0,1\} \times [0,1]$;
(2) and the second projection is injective on each component of the graph without its vertices, $\Gamma - \Gamma_0$; this makes it acyclic and progressive.

We call to the components of $\Gamma - \Gamma_0$ *wires*, W; we call the vertices of the graph *nodes*, Γ_0. Wires must be labelled by the objects of the polygraph, $o\colon W \to \mathcal{G}_{obj}$, nodes must be labelled by the generators of the polygraph, $m\colon \Gamma_0 \to \mathcal{G}$; and each node must be connected to wires exactly typed by the objects of its generator – a string diagram must be well-typed.

Lemma 1.1.10. *String diagrams over a polygraph $\mathcal{G}$ form a monoidal category, which we call* $\mathsf{String}(\mathcal{G})$. *This determines a functor,*

$$\mathsf{String}\colon \mathbf{PolyGraph} \to \mathbf{MonCat}_{\mathsf{Str}}.$$

Proof sketch. The objects of the category are lists of objects of the polygraph, which we write as $[X_0, \dots, X_n]$, for $X_i \in \mathcal{G}_{obj}$. These form a (free) monoid with concatenation and the empty list.

Morphisms $[X_0, \dots, X_n] \to [Y_0, \dots, Y_m]$ are string diagrams over the polygraph $\mathcal{G}$ such that *(i)* the ordered list of wires that touches the upper boundary is typed by $[X_0, \dots, X_n]$, and *(ii)* the ordered list of wires that touches the lower boundary is typed by $[Y_0, \dots, Y_m]$.

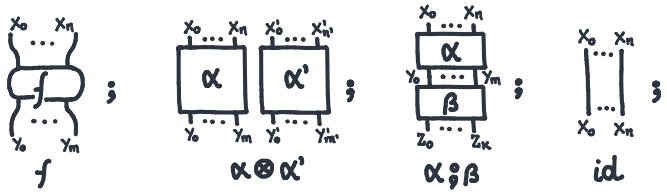


Figure 2. Strict monoidal category of string diagrams.

Figure 2 describes the operations of the category. The parallel composition of two diagrams $\alpha\colon [X_0, \dots, X_n] \to [Y_0, \dots, Y_m]$ and $\alpha'\colon [X'_0, \dots, X'_{n'}] \to [Y'_0, \dots, Y'_{m'}]$ is their horizontal juxtaposition. The sequential composition of two diagrams $\alpha\colon [X_0, \dots, X_n] \to [Y_0, \dots, Y_m]$ and $\beta\colon [Y_0, \dots, Y_m] \to [Z_0, \dots, Z_k]$ is the diagram obtained by vertical juxtaposition linking the outputs of the first to the inputs of the second. The identity on the object $[X_0, \dots, X_n]$ is given by a diagram containing n identity wires labelled by these objects. □

Lemma 1.1.11. *Forgetting about the sequential and parallel composition defines a functor from monoidal categories to polygraphs,*

$$\mathsf{Forget}\colon \mathbf{MonCat}_{\mathsf{Str}} \to \mathbf{PolyGraph}.$$

Proof. Any monoidal category $\mathbb{C}$ can be seen as a polygraph $\mathsf{Forget}(\mathbb{C})$ where the edges are determined by the morphisms,

$$\mathsf{Forget}(\mathbb{C})(A_0, \dots, A_n; B_0, \dots, B_m) = \mathbb{C}(A_0 \otimes \dots \otimes A_n, B_0 \otimes \dots \otimes B_m),$$

and we forget about composition and tensoring. It can be checked, by its definition, that any strict monoidal functor induces a homomorphism on the underlying polygraphs. □

Theorem 1.1.12 (Joyal and Street, [JS91, Theorem 2.3]). *There exists an adjunction between polygraphs and strict monoidal categories,* $\mathsf{String} \dashv \mathsf{Forget}$. *Given a*

polygraph $\mathcal{G}$, the free strict monoidal category $\mathsf{String}(\mathcal{G})$ *is the strict monoidal category that has as morphisms the string diagrams over the generators of the polygraph; the underlying polygraph determines the right adjoint.*

1.1.4. Example: Crema di Mascarpone. This first example shows how to construct morphisms in a monoidal category. The theory for preparing crema di mascarpone contains the following resources,

$$\{\mathsf{egg}, \mathsf{white}, \mathsf{yolk}, \mathsf{shell}, \mathsf{whisked\ white}, \mathsf{sugar}, \mathsf{mascarpone}, \mathsf{paste}, \mathsf{thick\ paste}, \mathsf{crema}\}.$$

These resources are the objects of the polygraph also containing the following seven generators, as in Figure 3.

(1) crack: $\mathsf{egg} \to \mathsf{white} \otimes \mathsf{shell} \otimes \mathsf{yolk}$,
(2) beat: $\mathsf{yolk} \otimes \mathsf{yolk} \otimes \mathsf{sugar} \to \mathsf{paste}$,
(3) stir: $\mathsf{paste} \otimes \mathsf{mascarpone} \to \mathsf{thick\ paste}$,
(4) whisk: $\mathsf{white} \otimes \mathsf{white} \to \mathsf{whisked\ whites}$,
(5) fold: $\mathsf{whisked\ whites} \otimes \mathsf{thick\ paste} \to \mathsf{cream}$,
(6) swap: $\mathsf{yolk} \otimes \mathsf{white} \to \mathsf{white} \otimes \mathsf{yolk}$,
(7) discard: $\mathsf{shell} \to \mathrm{I}$.

All these resources form a polygraph $\mathcal{C}$. Thanks to the adjunction between polygraphs and monoidal categories, deciding how to interpret the resources and the generators of the polygraph in any monoidal category, $\mathcal{C} \to \mathsf{Forget}(\mathbb{D})$, is the same as creating a strict monoidal functor that interprets string diagrams in that category, $\mathsf{String}(\mathcal{C}) \to \mathbb{D}$. In particular, it interprets Figure 1 as a morphism in the monoidal category $\mathbb{D}$.

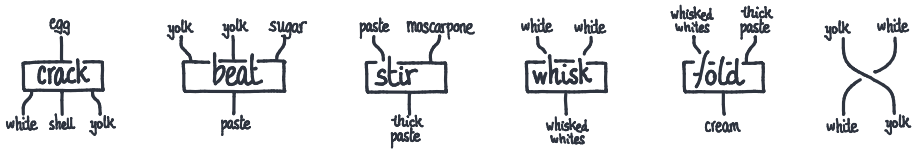


FIGURE 3. Polygraph for the theory of mascarpone.

Usually, discarding and swapping are better understood as global structure with which all of the objects of the category are endowed. Even better than asking for the last generator in Figure 3, we will ask for the ability to copy or to swap resources freely. This is done via cartesian monoidal categories or symmetric monoidal categories, respectively.

1.1.5. Bibliography. Monoidal categories, together with their coherence theorem, were first introduced by MacLane [Mac63, Mac78], explicit equivalence theorems are given later by Joyal and Street [JS93]. String diagrams as progressive graphs were introduced by Joyal [JS91]. Our presentation follows the *resource theories* of Coecke, Fritz and Spekkens [CFS16].

Petri nets and process calculi are alternative mathematical approaches to process theory; at the same time, they are arguably particular cases of monoidal categories [Sob10]. Monoidal categories as circuits and processes are explicitly pioneered by Sabadini, Walters, Carboni and Street [KSW97, CW87]. The string diagrammatic recipe for "crema di mascarpone" is an adaptation of a blog post by Sobocinski [Sob13]. We also follow the ideas of Shulman on categorical logic [Shu16].

1.2. Non-Strict Monoidal Categories

We have just argued for the axioms of strict monoidal categories as our process theories and string diagrams as our syntax. The bad news is that there exists a technicality preventing many interesting examples from ever forming a strict monoidal category; the good news is that neither the axioms of monoidal categories nor our string diagrams need to change at all to accommodate this technicality: this curious phenomenon is possible thanks to MacLane's *strictification and coherence* results [ML71]. This section introduces non-strict, or general monoidal categories, and it immediately details how MacLane's results ratify the axioms of strict monoidal categories.

1.2.1. Non-Strictness. The axioms of strict monoidal categories are enough to study a broader class of mathematical structures: non-strict monoidal categories.

What is a non-strict monoidal category? In a monoidal category, the tensor $(\otimes)$ is not required to be associative or unital. Because of how we usually construct our definitions, it is not always the case that $X \otimes (Y \otimes Z) = (X \otimes Y) \otimes Z$. Consider the theory of sets and functions, with the cartesian product $(\times)$ as the tensor. If we define

$$X \times Y = \{(x, y) \mid x \in X, y \in Y\},$$

then, simply, $X \times (Y \times Z) \neq (X \times Y) \times Z$. However, it is still the case that there exist two functions $X \times (Y \times Z) \to (X \times Y) \times Z$ and $(X \times Y) \times Z \to X \times (Y \times Z)$ and that these functions are mutual inverses for sequential composition. Whenever this happens in a category, we say these two objects are *isomorphic* $(\cong)$ and the morphisms are *isomorphisms*. In a monoidal category, the tensor is not associative and unital but it is still associative and unital up to an isomorphism. This may sound like a minor technicality, but it makes many examples fail to form a monoidal category.

Definition 1.2.1. A **monoidal category**, $(\mathbb{C}, \otimes, I, \alpha, \lambda, \rho)$, is a category $\mathbb{C}$ equipped with a pair of functors $(\otimes) \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$, and $I \in \mathbb{C}$, called tensor and unit respectively. and three families of isomorphisms called the *coherence maps*:

(1) the associator $\alpha_{X,Y,Z} \colon (X \otimes Y) \otimes Z \cong X \otimes (Y \otimes Z)$,
(2) the left unitor $\lambda_A \colon I \otimes A \cong A$ and
(3) the right unitor $\rho_A \colon A \otimes I \cong A$.

These three families of maps must be natural, meaning they commute with other well-typed morphisms of the monoidal category. Moreover, these must be such that every formally well-typed equation between coherence maps holds.

Proposition 1.2.2. *A strict monoidal category is precisely a monoidal category where α, λ, and ρ are identities.*

Proof. The naturality of the coherence maps, whenever these are identities, is the same as the associativity and unitality of the tensor (3 and 4). Functoriality of the tensor is the same as the interchange axioms (5 and 6); while the functoriality of the unit is trivially true. □

1.2.2. Coherence. The conditions of the definition of a non-strict monoidal category may seem too strong: we are asking that a wide family of equations (all the formally well-typed ones) hold. Fortunately, the *coherence theorem* shows that simply checking two families of equations is enough.

Theorem 1.2.3 (MacLane, [ML71]). *The three different families of coherence maps (α, λ, ρ) satisfy all formally well-typed equations between them whenever they satisfy the* triangle *and* pentagon *equations:*

(1) $\alpha_{X,I,Y} \fatsemi (\mathrm{id}_X \otimes \lambda_Y) = \rho_X \otimes \mathrm{id}_Y$, *and*
(2) $(\alpha_{X,Y,Z} \otimes \mathrm{id}) \fatsemi \alpha_{X,Y\otimes Z,W} \fatsemi (\mathrm{id}_X \otimes \alpha_{Y,Z,W}) = \alpha_{X\otimes Y,Z,W} \fatsemi \alpha_{X,Y,Z\otimes W}$.

Finally, the theorem that allows strict monoidal categories to talk about non-strict monoidal categories is the *strictification theorem* that says that any monoidal category is monoidally equivalent to a strict one. This does determine an adjunction with extra structure, making it a 2-adjunction [Cam19].

THEOREM 1.2.4 (Joyal and Street, [JS91]). *Any monoidal category is equivalent via a strong monoidal functor to a strict one. There exists a 2-adjunction between the category of monoidal categories and strong monoidal functors and the category of strict monoidal categories with strict monoidal functors. Moreover, the unit of this 2-adjunction is an equivalence.*

1.2.3. String Diagrams of Monoidal Categories. While it is true that we can construct the free strict monoidal category on a polygraph, it is not true yet that we know how to construct the free monoidal category (the non-strict one) over a polygraph; in fact, this seems to be impossible. This could be misread as saying that string diagrams are not equally sound and complete for monoidal categories. Nothing is further from the truth: even if there are now multiple ways of interpreting a string diagram in a monoidal category, these are essentially equal – they define isomorphic functors.

THEOREM 1.2.5. *There is a pseudoadjunction between the locally discrete 2-category of polygraphs and the 2-category of monoidal categories, strong monoidal functors and monoidal natural transformations.*

PROOF SKETCH. This pseudoadjunction arises as a combination of two different 2-adjunctions. The first one is the adjunction between polygraphs and strict monoidal categories we have studied before. The second one is described in Theorem 1.2.4, and its unit defines an equivalence: every monoidal category is equivalent to a strict one.

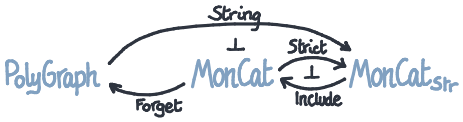


FIGURE 4. Pseudoadjunctions between polygraphs and monoidal categories.

It is well known that each time that we have two adjunctions in this disposition we can reduce one along the other, provided that the unit of the former is invertible (see Proposition 1.3.10). In this case, the unit is not exactly invertible but merely an equivalence: as a consequence, we obtain not another 2-adjunction but merely a pseudoadjunction. This concludes the proof. □

1.2.4. Bibliography. The coherence results go back to MacLane, Joyal and Street [Mac63, JS91]; Hermida arrived at the same result via multicategories [Her01, Had18]; and there is a more modern account by Becerra detailing the 2-adjunction between strict and non-strict monoidal categories [Bec23], which Campbell studies for bicategories [Cam19].

1.3. String Diagrams of Bicategories

Bicategories are the second extension of monoidal categories that we will employ during the text. If monoidal categories were well-suited to reason about process theories, bicategories, one level up, are well-suited to reason about categories themselves; however, their string diagrammatic syntaxes are very close.

String diagrams of monoidal categories can be easily extended to bicategories if we allow ourselves to color the regions. Coloring the regions simply constrains which objects can be tensored: this algebraic structure is a bicategory, also known as a *weak 2-category*. In a bicategory, two objects must coincide along a boundary to be tensored.

Definition 1.3.1. A strict *2-category* $\mathbb{B}$ consists of a collection of *objects*, or 0-cells, $\mathbb{B}_{obj}$, and a category of *morphisms* or 1-cells between any two objects, $\mathbb{B}(A;B)$. A strict *2-category* is endowed with operations for the parallel composition of 1-cells,

$$(\mathbin{;})\colon \mathbb{B}(A;B) \times \mathbb{B}(B;C) \to \mathbb{B}(A;C),$$
$$(I_A)\colon \mathbb{B}(A;A),$$

that are associative and unital both on objects and morphisms, meaning that $(X \mathbin{;} Y) \mathbin{;} Z = X \mathbin{;} (Y \mathbin{;} Z)$, and $I_A \mathbin{;} X = X = X \mathbin{;} I_B$. Bicategories must satisfy the following axioms, making parallel composition a functor:

(1) parallel composition is unital, $f \mathbin{;} \mathrm{id} = f$, and $\mathrm{id} \mathbin{;} f = f$;
(2) parallel composition is associative, $f \mathbin{;} (g \mathbin{;} h) = (f \mathbin{;} g) \mathbin{;} h$;
(3) compositions are unital, $\mathrm{id} \mathbin{;} \mathrm{id} = \mathrm{id}$;
(4) compositions interchange, $(f \fatsemi g) \mathbin{;} (f' \fatsemi g') = (f \mathbin{;} f') \fatsemi (g \mathbin{;} g')$.

Remark 1.3.2. A single-object strict 2-category is exactly a strict monoidal category.

1.3.1. String diagrams of 2-categories. Let us briefly comment on how the string diagrams of monoidal categories extend to bicategories. We repeat the same definitions and the same theorems, just taking care of matching the boundaries this time.

Definition 1.3.3. A *bigraph*, or *2-graph*, $\mathcal{B}$ is given by a set of objects, $\mathcal{B}_{obj}$; a set of arrows between any two objects, $\mathcal{B}(A;B)$; and a set of 2-arrows between any two paths of arrows, $\mathcal{B}(X_0,\dots,X_n;Y_0,\dots,Y_m)$.

A bigraph homomorphism, $f\colon \mathcal{A} \to \mathcal{B}$, is a function between their object sets, $f_o\colon \mathcal{A}_{obj} \to \mathcal{B}_{obj}$; a family of functions between their corresponding arrow sets, $f\colon \mathcal{A}(A;B) \to \mathcal{B}(f(A),f(B))$; and a family of functions between their corresponding 2-arrow sets,

$$f\colon \mathcal{A}(X_0,\dots,X_n;Y_0,\dots,Y_m) \to \mathcal{B}(fX_0,\dots,fX_n;fY_0,\dots,fY_m).$$

Bigraphs with bigraph homomorphisms form a category, **BiGraph**.

Definition 1.3.4. A string diagram over a bigraph $\mathcal{A}$ is a string diagram over the polygraph formed by arrows and 2-arrows, additionally satisfying that each region is labelled by an object of the bigraph, and in such a way that any wire is labelled by an arrow connecting the labels of the two regions.

THEOREM 1.3.5. *There is an adjunction between bicategorical graphs and 2-categories with strict 2-functors between them. The left adjoint is given by colored string diagrams over the bigraph.*

1.3.2. Bicategories. Strict monoidal categories have a weak analogue that still shares the same syntax – (weak, or non-strict) monoidal categories. In the same way, strict 2-categories have a weak analogue that shares the same syntax: weak 2-categories, sometimes called bicategories.

Definition 1.3.6. A *bicategory* $(\mathbb{B}, ;, I, \alpha, \lambda, \rho)$ is a collection of 0-cells, $\mathbb{B}_{obj}$, together with a category $\mathbb{B}(A;B)$ between any two 0-cells, $A, B \in \mathbb{B}_{obj}$, and functors

$$(;)\colon \mathbb{B}(A;B) \times \mathbb{B}(B;C) \to \mathbb{B}(A;C), \text{ and } I_A\colon \mathbb{B}(A;A),$$

that are associative and unital up to isomorphism, meaning that there exist natural isomorphisms describing associativity $\alpha_{X,Y,Z}\colon (X \otimes Y) \otimes Z \cong X \otimes (Y \otimes Z)$, left unitality $\lambda_X\colon I_A \otimes X \cong X$ and right unitality $\rho_X\colon X \otimes I_B \cong B$.

Conjecture 1.3.7. *There is a pseudoadjunction between the locally discrete 2-category of bicategorical graphs and the 2-category of bicategories, pseudofunctors and icons (see Campbell, Garner and Gurski's work for the higher structure of the strictification adjunction [Cam19, GG09]).*

1.3.3. Example: Adjunctions. We exemplify the usage of string diagrams for bicategories in an abstract definition of adjunctions. We then use string diagrams to prove a theorem about adjunctions.

Definition 1.3.8. The *theory of a duality* in a bicategory contains two 0-cells A and B; it contains two 1-cells between them, $L\colon A \to B$ and $R\colon B \to A$, and it contains two 2-cells, $\varepsilon\colon L \fatsemi R \to I$ and $\eta\colon I \to R \fatsemi L$, that satisfy $(\mathrm{id} \otimes \eta) \fatsemi (\varepsilon \otimes \mathrm{id}) = \mathrm{id}$ and $(\eta \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes \varepsilon) = \mathrm{id}$.

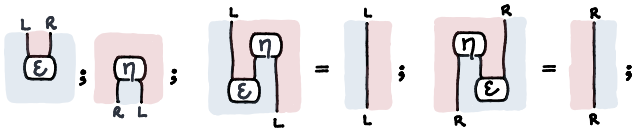


FIGURE 5. Theory of a duality.

Remark 1.3.9. Adjunctions are dualities in the bicategory of categories, functors, and natural transformations.

Proposition 1.3.10 (Reducing an adjunction)**.** *Let $F\colon \mathbb{A} \to \mathbb{C}$ and $H \fatsemi U\colon \mathbb{C} \to \mathbb{A}$ determine an adjunction $(F, H \fatsemi U, \eta, \varepsilon)$ and let $P\colon \mathbb{B} \to \mathbb{C}$ determine a second adjunction (P, H, u, c) such that the unit $u\colon I \to P \fatsemi H$ is a natural isomorphism (as in Figure 6). Then, $F \fatsemi H$ is left adjoint to U.*

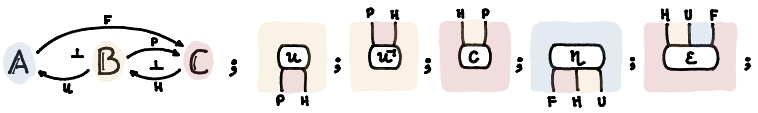


FIGURE 6. Setting for reducing an adjunction.

PROOF. We employ the string diagrammatic calculus of bicategories for the bicategory of categories, functors and natural transformations [Mar14]. We define the morphisms in Figure 7 to be the unit and the counit of the adjunction. We then prove that they satisfy the snake equations in Figures 7 and 8.

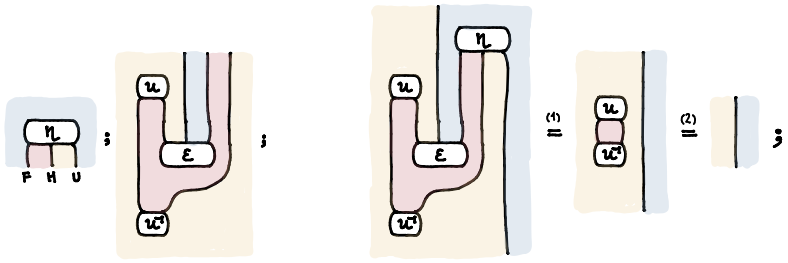


FIGURE 7. Unit and counit of the reduced adjunction (left). First snake equation (right).

In the first snake equation, in Figure 7, we use *(i)* that there is a duality (η, ε), and *(ii)* that u is invertible. In the second snake equation, in Figure 8, we use *(i)* that there is a duality (u, c), *(ii)* that u is invertible, *(iii)* that there is a duality (u, c), again; and *(iv)* that there is a duality (η, ε). □

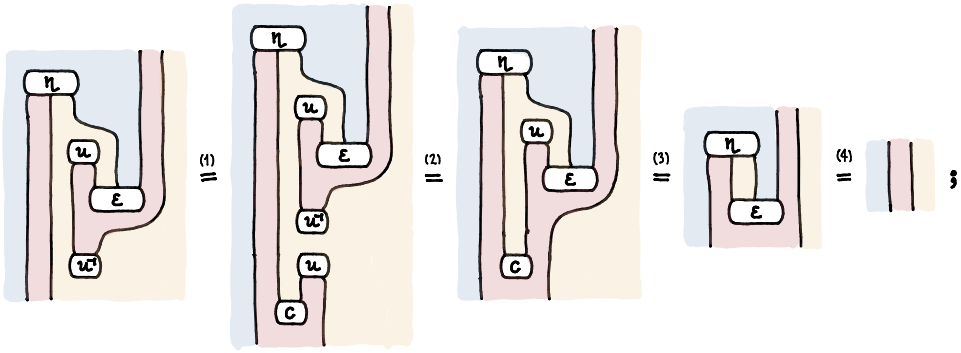


FIGURE 8. Second snake equation.

1.3.4. Bibliography. String diagrams for bicategories are usually interpreted as the Poincaré dual of *globular pasting diagrams*, which were used by Bénabou since the introduction of bicategories [Bén67]. Marsden uses string diagrams for bicategories to study basic formal category theory [Mar14].

1.4. Symmetric Monoidal Categories and Do-Notation

Setting strictness asside, an extra axiom sharpens monoidal categories for the study of process theories: *symmetry*. Symmetry states that the position that resources occupy is not important: $A \otimes B$ is worth the same as $B \otimes A$. Assuming symmetry simplifies process syntax drammatically – in fact, it also enables a new syntax that mimics imperative programming – and it is arguably an axiom that we need before we can really talk of *processes*.

This section introduces symmetric monoidal categories, their specialized string diagrams in terms of hypergraphs [BGK+16], and their do-notation syntax [Hug00].

1.4.1. Commutative Monoidal Categories. So far, the meaning of the tensor ($\otimes$) in process theories has arguably been too general: so far, we have asked $A \otimes B$ to represent the juxtaposition of resources – resources form a monoid. Why not a commutative monoid? If we interpret objects as bags of resources, it seems clear that a commutative monoid would be more appropriate. However, imposing commutativity naively fails catastrophically: the individuality of each resource disappears [MM90].

Definition 1.4.1. A *commutative monoidal category* is a strict monoidal category $(\mathbb{C}, \otimes, I)$ where objects form a commutative monoid, $A \otimes A' = A' \otimes A$ for each $A, A' \in \mathbb{C}_{obj}$, and the tensor of morphisms is also commutative, $f \otimes f' = f' \otimes f$ for any two $f\colon A \to B$ and $f'\colon A' \to B'$.

Proposition 1.4.2. *In a commutative monoidal category, resources do not have individuality: it does not matter to which of them we apply a transformation, and not even the order in which we apply them. More formally,*

$$(f \fatsemi g) \otimes \mathrm{id} = (f \otimes g) = \mathrm{id} \otimes (g \fatsemi f)$$

for each two transformations of the same resource, $f\colon X \to X$ and $g\colon X \to X$.

This may be useful in specific applications. Indeed, it is one of the crucial ideas behind the formalization of Petri nets as monoids in the work of Meseguer and Montanari [MM90] – commutativity represents the "collective token philosophy" of Petri nets [BGMS21]. However, for our purposes, we will need a more refined notion that does not destroy the individuality of our resources: this notion is given by symmetric monoidal categories.

1.4.2. Symmetric Monoidal Categories. *Symmetric monoidal categories* do not assume that the monoid of objects is commutative; they only assume that there is a family of processes that allow us to reorder resources, making it commutative "up to an invertible process". In practice, this means that even when $X \otimes Y \neq Y \otimes X$, there exists a process $\sigma_{X,Y}\colon X \otimes Y \to Y \otimes X$ that is invertible. These are our definitive notion of *process theory*.

Definition 1.4.3. A strict *symmetric monoidal category* (or a *permutative category*) is a strict monoidal category $(\mathbb{C}, \otimes, I)$ endowed with a family of maps $\sigma_{X,Y}\colon X \otimes Y \to Y \otimes X$ that satisfy the following equations describing how to

(1) swap nothing, $\sigma_{I,X} = \mathrm{id}_X = \sigma_{X,I}$;
(2) swap resources on the left, $\sigma_{X,Y\otimes Z} = (\sigma_{X,Y} \otimes \mathrm{id}_Z) \fatsemi (\mathrm{id}_Y \otimes \sigma_{X,Z})$;
(3) swap resources on the right, $\sigma_{X\otimes Y,Z} = (\mathrm{id}_X \otimes \sigma_{Y,Z}) \fatsemi (\sigma_{X,Z} \otimes \mathrm{id}_Y)$;
(4) reverse a swap with a swap, $\sigma_{X,Y} \fatsemi \sigma_{Y,X} = \mathrm{id}_X \otimes \mathrm{id}_Y$;
(5) swap and apply transformations, $\sigma_{X,Y} \fatsemi (f \otimes g) = (g \otimes f) \fatsemi \sigma_{X',Y'}$;
(6) and swap in any order,

$$(\sigma_{X,Y} \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes \sigma_{X,Z}) \fatsemi (\sigma_{Y,Z} \otimes \mathrm{id}) = (\mathrm{id} \otimes \sigma_{Y,Z}) \fatsemi (\sigma_{X,Z} \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes \sigma_{X,Y}).$$

The first three axioms are especially important for clarifying all the rest: they say that the swapping process of any two objects in a freely generated monoidal category is determined by the swapping process of the generators. This already allow us to have a first string diagrammatic calculus for symmetric monoidal categories: the swap on the generators is represented by wires crossing; the swap on arbitrary objects is constructed from it; the rest of the axioms are better understood in terms of string diagrams (Figure 9).

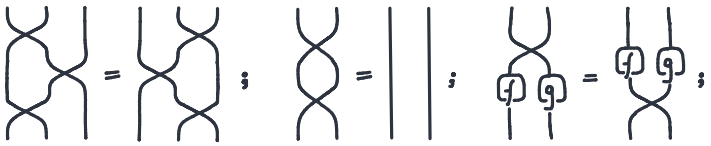


FIGURE 9. Theory of strict symmetric monoidal categories.

However, this is an inefficient syntax: it forces us to explicitly deal with the axioms of the swap. A better syntax would make them transparent, and state that the only thing that matters in a string diagram for symmetric monoidal categories is where the wires are ultimately connected – that is, we only care about the underlying hypergraph. A detailed presentation of the string diagrams of symmetric monoidal categories as hypergraphs is in the work of Bonchi, Sobocinski, Zanasi, and others [BSZ14, BGK+16].

Definition 1.4.4. A *hypergraph* (V, E) consists of a set of nodes, V, and a set of directed hyperedges E connecting lists of vertices to lists of vertices, that is, $e\colon [v_1, \dots, v_n] \to [w_1, \dots w_m]$ for each $e \in E$. We say a hypergraph is acylic if contains no loops.

Definition 1.4.5. A *hypergraph labelled over a polygraph* $\mathcal{G}$, is a hypergraph (V, E) such that each vertex $v \in V$ is assigned an object of the polygraph, $l(v) \in \mathcal{G}_{obj}$, and each hyperedge $e\colon [v_1, \dots, v_n] \to [w_1, \dots w_m]$ is assigned an edge

$$l(e)\colon l(v_1), \dots, l(v_n) \to l(w_1), \dots, l(w_m),$$

preserving the type of its vertices.

Definition 1.4.6. A *symmetric string diagram* from $[X_1, \dots, X_n]$ to $[Y_1, \dots, Y_m]$ is an acyclic hypergraph (V, E) labelled by a polygraph $\mathcal{G}$, such that each vertex appears exactly once as an input and exactly once as an output, and endowed with two distinguished unlabelled hyperedges:

$$\text{the input } i\colon [] \to [x_1, \dots, x_n], \text{ and the output } o\colon [y_1, \dots, y_m] \to [],$$

typed by $l(x_1) = X_1, \dots, l(x_n) = X_n$ and $l(y_1) = Y_1, \dots, l(y_m) = Y_m$.

Proposition 1.4.7. *Symmetric string diagrams over a polygraph* $\mathcal{G}$ *form a symmetric monoidal category,* $\mathsf{String}_\sigma(\mathcal{G})$.

PROOF. A summary of the construction is in Figure 10: wires are vertices and hyperedges are nodes. Let us describe the category. The objects are lists of objects of the polygraph. Tensoring of string diagrams of type $[X_1, ..., X_n] \to [Y_1, ..., Y_m]$ and $[X'_1, ..., X'_n] \to [Y'_1, ..., Y'_m]$ is defined by the disjoint union of the hypergraphs – merging input and output edges – into a string diagram $[X_1, ..., X_n, X'_1, ..., X'_n] \to [Y_1, ..., Y_m, Y'_1, ..., Y'_m]$. Composition of string diagrams of type $[X_1, ..., X_n] \to [Y_1, ..., Y_m]$ and $[Y_1, ..., Y_m] \to [Z_1, ..., Z_p]$ is constructed by glueing the vertices along the output edge of the first and the input edge of the second, which disappear

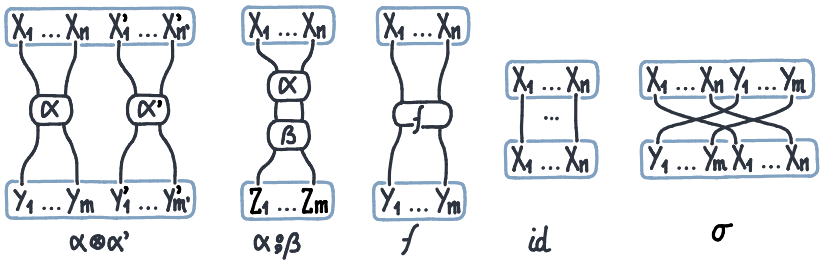


FIGURE 10. Symmetric monoidal category of string diagrams.

producing a string diagram $[X_1, \ldots, X_n] \to [Z_1, \ldots, Z_p]$. Generators are included as single hyperedges labelled by them. The identity, $[X_1, \ldots, X_n] \to [X_1, \ldots, X_n]$, consists only of the input and output edges, connected by a list of vertices. Symmetries are defined by twisting the connections of the input and output edges. Finally, we can check that string diagrams satisfy the axioms of symmetric monoidal categories. □

Definition 1.4.8. A *strict symmetric monoidal functor* between two strict symmetric monoidal categories, $F\colon \mathbb{C} \to \mathbb{D}$, is a strict monoidal functor that moreover preserves symmetries: $F(\sigma)\colon F(X) \otimes F(Y) \to F(Y) \otimes F(X)$ is the symmetry on $F(X)$ and $F(Y)$. Strict symmetric monoidal categories and strict symmetric monoidal functors form a category, $\mathbf{SymMonCat}_{\mathsf{Str}}$.

Proposition 1.4.9. *The construction of symmetric string diagrams extends to a functor from polygraphs to symmetric monoidal categories,*

$$\mathsf{String}_\sigma : \mathbf{PolyGraph} \to \mathbf{SymMonCat}_{\mathsf{Str}}.$$

Proposition 1.4.10. *There exists a forgetful functor from strict symmetric monoidal categories to polygraphs that takes the objects as the vertices of the polygraph and the morphisms as the edges,*

$$\mathsf{Forget} : \mathbf{SymMonCat}_{\mathsf{Str}} \to \mathbf{PolyGraph}.$$

THEOREM 1.4.11. *String diagrams for symmetric monoidal categories form the free strict symmetric monoidal category over a polygraph: there exists an adjunction* $\mathsf{String}_\sigma \dashv \mathsf{Forget}$.

PROOF SKETCH. We have already shown that string diagrams form a symmetric monoidal category. It only remains to show that there exists a unique strict symmetric monoidal functor to any strict symmetric monoidal category. The assignment is determined by the fact that each string diagram is constructed from the generators and the operations of a symmetric monoidal category, the difficulty is in showing that this assignment is well-defined. We refer the reader to the work of Bonchi and others [BGK+16]. □

The syntax of symmetric string diagrams as hypergraphs is more efficient: to check equality, only the connectivity of the wires matters, and we no longer need to track the specific blocks forming the diagram.

1.4.3. Do-Notation. There is a second practical syntax for symmetric monoidal categories that links string diagrams to programming: Hughes' *arrow do-notation* [Hug00, Pat01]. It comes from *functional programming*, but it is precisely a representation of *imperative programming*. The main idea is that, in a string diagram, we can label the wires by variable names, and simply declare which nodes

take which inputs and outputs to reconstruct the string diagram. In a certain sense, this is the graph encoding of a string diagram, but it closely resembles an imperative program.

Example 1.4.12 (Crema di Mascarpone in Do-notation)**.** Consider the same process for "crema di mascarpone" that we detailed in Section 1.1.4. This time, we can directly assume that we are in a symmetric monoidal category. The translation of the string diagram of Figure 1 is the following code in Figure 11.

```
cremaDiMascarpone(egg1, egg2, sugar, mascarpone):
  crack(egg1) -> (white1, shell1, yolk1)
  crack(egg2) -> (white2, shell2, yolk2)
  whisk(white1, white2) -> whiskedWhites
  beat(yolk1, yolk2, sugar) -> paste
  stir(paste, mascarpone) -> thinkPaste
  fold(whiskedWhites, thickPaste) -> crema
  return(crema)
```

FIGURE 11. Do-notation recipe for Crema di Mascarpone.

Let us formalize do-notation in the style of a type-theory: we will work with variables, we consider them to be unique and we work implicitly up to α-equivalence (or renaming of variables). Our notion of signature is again that of a polygraph: our basic types will be the objects of the polygraph, and we will have a rule for each one of the generators in the polygraph.

Definition 1.4.13. A *derivation* in the proof theory of do-notation over a polygraph is defined inductively to be either

(1) a single return statement, given by a permutation; or
(2) the application of a generator $f\colon A_0, \dots, A_n \to B_0, \dots, B_m$, given by a choice of generator and an insertion of variables, followed by a derivation.

$$\frac{}{\mathsf{a}_0{:}A_0, \dots, \mathsf{a}_n{:}A_n \gg_\tau () \vdash \mathsf{return}(\mathsf{a}_0, \dots, \mathsf{a}_n) : A_0 \otimes \dots \otimes A_n}\ (\textsc{Return})$$

$$\frac{\mathsf{b}_0{:}B_0, \dots, \mathsf{b}_m{:}B_m, \Gamma \vdash \mathsf{t} : \Delta}{\mathsf{a}_0{:}A_0, \dots, \mathsf{a}_n{:}A_n \gg_\tau \Gamma \vdash \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to \mathsf{b}_0, \dots, \mathsf{b}_m \fatsemi \mathsf{t} : \Delta}\ (f)$$

FIGURE 12. Do-notation for symmetric monoidal categories.

Before continuing, then, it is important to understand what an *insertion* is. An insertion captures how many ways we have of inserting some n new terms into a list of m terms. The new n terms can be freely permuted, but the list of m terms must preserve their relative order. This is the combinatorial structure that will track how variables are used in a derivation.

Definition 1.4.14. We define the family of insertions of n terms into m terms, $\mathsf{Ins}(n, m)$, inductively. There exists a single way of inserting zero terms into any list of terms, $\mathsf{Ins}(0, m) = 1$; inserting $n + 1$ terms into a list of m terms amounts to choosing the position of the first among $m + 1$ possible choices, and inserting the rest of the terms,

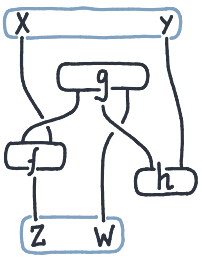
$$\mathsf{Ins}(n + 1, m) = (m + 1) \times \mathsf{Ins}(n, m + 1).$$

As a consequence, the number of possible insertions is $\mathsf{Ins}(n, m) = (m + n)!/m!$.

We write $a_1, \ldots, a_n \gg_\tau \Gamma$ to refer to the list resulting from the insertion of the variables $a_1, \ldots, a_n$ into the list Γ, of length m, according to the insertion $\tau \in \mathsf{Ins}(n, m)$.

Remark 1.4.15. Accordingly, the only information that a return statement may carry is that of an insertion $\mathsf{Ins}(n, 0)$, which is equivalently a permutation of the n elements that are being returned. The information carried by a generator statement is not only a generator n-to-m but also an insertion $\mathsf{Ins}(n, \#\Gamma)$ of n variables on the context of the derivation.

Example 1.4.16. Let us provide an example of the correspondence between the different notations: a Rosetta's stone translating between string diagrams, terms of do-notation, and their corresponding derivations (Figure 13).



$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{z}{:}Z, \mathsf{w}{:}W \vdash \mathsf{return}(\mathsf{z}, \mathsf{w}) : Z \otimes W}}{\mathsf{z}{:}Z, \mathsf{b}{:}B, \mathsf{w}{:}W, \mathsf{y}{:}Y \vdash \mathsf{h}(\mathsf{b}, \mathsf{y}) \to () \fatsemi \mathsf{return}(\mathsf{z}, \mathsf{w}) : Z \otimes W}}{\mathsf{a}{:}A, \mathsf{b}{:}B, \mathsf{w}{:}W, \mathsf{x}{:}X, \mathsf{y}{:}Y \vdash \mathsf{f}(\mathsf{a}, \mathsf{x}) \to \mathsf{z} \fatsemi \mathsf{h}(\mathsf{b}, \mathsf{y}) \to () \fatsemi \mathsf{return}(\mathsf{z}, \mathsf{w}) : Z \otimes W}}{\mathsf{x}{:}X, \mathsf{y}{:}Y \vdash \mathsf{g}() \to (\mathsf{a}, \mathsf{b}, \mathsf{w}) \fatsemi \mathsf{f}(\mathsf{a}, \mathsf{x}) \to \mathsf{z} \fatsemi \mathsf{h}(\mathsf{b}, \mathsf{y}) \to () \fatsemi \mathsf{return}(z, w) : Z \otimes W}$$

FIGURE 13. String diagram and derivation of a term.

1.4.4. Symmetry in Do-notation. At this point, using insertions may seem complicated: why not simply assume an exchange rule that allows us to permute variables freely? The problem we would encounter is that exchanges introduce redundancy: there would be multiple ways of writing the same term, depending on where we place the symmetries (Shulman describes the same problem for a different notation [Shu16]). This is not a catastrophic problem – we could still quotient them out appropriately – but it would make the construction much more complicated than simply dealing with the combinatorial structure of insertions upfront.

The better solution is to have exchange appear as a derived, admissible rule, rather than a primitive.

Proposition 1.4.17. *Exchange is admissible in Do-notation for symmetric monoidal categories. Any derivation* $\Gamma, \mathsf{x}, \mathsf{y}, \Delta \vdash \mathsf{t} : X$ *admits a derivation* $\Gamma, \mathsf{y}, \mathsf{x}, \Delta \vdash \mathsf{t} : X$.

PROOF. We proceed by structural induction. The base case is a single return statement, written as $\mathsf{a}_0, \ldots, \mathsf{x}, \mathsf{y}, \ldots, \mathsf{a}_n \gg_\sigma () \vdash \mathsf{return}(\mathsf{a}_0, \ldots, \mathsf{x}, \ldots, \mathsf{y}, \ldots \mathsf{a}_n) : X$. Permuting x, y in the insertion τ gives us a new insertion τ_{xy} deriving the same statement under a different context

$$\mathsf{a}_0, \ldots, \mathsf{y}, \mathsf{x}, \ldots, \mathsf{a}_n \gg_{\tau_{\mathsf{xy}}} () \vdash \mathsf{return}(\mathsf{a}_0, \ldots, \mathsf{x}, \ldots, \mathsf{y}, \ldots \mathsf{a}_n) : X.$$

Consider now an application of a generator,

$$\mathsf{a}_0, \ldots, \mathsf{a}_n \gg_\tau \Gamma \vdash \mathsf{f}(\mathsf{a}_0, \ldots, \mathsf{a}_n) \to \mathsf{b}_0, \ldots, \mathsf{b}_m \fatsemi \mathsf{term} : X.$$

There are two possible cases here: (1) if any of x, y is among the inserted variables, $\mathsf{a}_0, \ldots, \mathsf{a}_n$, then we may simply exchange them by changing the order in which they

are inserted; (2) if not, then $\Gamma = \Gamma', \mathsf{x}, \mathsf{y}, \Delta'$, and we apply the induction hypothesis over the derivation $\Gamma', \mathsf{x}, \mathsf{y}, \Delta' \vdash \mathsf{term} : X$. □

Indeed, this is enough to ensure that terms do correspond to derivations: we may simply write a term and there is a unique way it could have been extracted from the context.

Proposition 1.4.18. *Do-notation terms in a given context have a unique derivation.*

Proof. By structural induction, a term is either a single return or an application of some generator (f). Any single term $\mathsf{x}_0, \dots, \mathsf{x}_n \vdash \mathsf{return}(\mathsf{a}_0, \dots, \mathsf{a}_n)$ has a unique derivation: namely, the one that inserts the a's into the x's permuting them in the only possible order. The insertion τ must be the only one making $\tau(\mathsf{x}_i) = \mathsf{a}_i$.

Consider now an application of a generator, $\Delta \vdash \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to \mathsf{b}_0, \dots, \mathsf{b}_m \fatsemi \mathsf{t} : X$. We know that it must come from $\Delta = (\mathsf{a}_0, \dots, \mathsf{a}_n \gg_\tau \Gamma)$, and this forces $\Gamma = (\Delta - \{\mathsf{a}_0, \dots, \mathsf{a}_n\})$ and the value of the insertion τ: it is the only one that turns Γ into Δ. Once Γ has been determined, we apply the induction hypothesis to get the derivation of $\mathsf{b}_0, \dots, \mathsf{b}_m, \Gamma \vdash \mathsf{t} : X$. □

1.4.5. Quotienting Do-notation. Our logic is freely constructed, but it is not yet a logic of monoidal categories: technically, it misses interchange. As it stands, it is actually a logic for symmetric premonoidal categories, which we will study later. We shall only add the following rule – the interchange rule – in order to convert it into a calculus of symmetric monoidal categories.

Interchange rule. Consider a derivation of the following term, where $\mathsf{b}_0, \dots, \mathsf{b}_m$ and $\mathsf{c}_0, \dots \mathsf{c}_p$ are two lists of *distinct* variables, meaning that no b_i appears in c_j – and conversely, no d_i appears in a_j. Then, we can interchange the two first statements of the term.

$$\begin{array}{l} \Gamma \vdash \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to \mathsf{b}_0, \dots, \mathsf{b}_m \fatsemi \\ \quad \mathsf{g}(\mathsf{c}_0, \dots, \mathsf{c}_p) \to \mathsf{d}_0, \dots, \mathsf{d}_p \fatsemi \\ \quad \mathsf{term} : \Delta \end{array} \equiv \begin{array}{l} \Gamma \vdash \mathsf{g}(\mathsf{c}_0, \dots, \mathsf{c}_p) \to \mathsf{d}_0, \dots, \mathsf{d}_p \fatsemi \\ \quad \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to \mathsf{b}_0, \dots, \mathsf{b}_m \fatsemi \\ \quad \mathsf{term} : \Delta \end{array}$$

Figure 14. Interchange rule.

Proposition 1.4.19. *We show this is well defined. Whenever the left hand side of the interchange rule is a valid term, and variables are distinct, the right hand side is also a valid term.*

Proof. First, we reason that the term on the left-hand side must have a derivation tree of the following form.

$$\dfrac{\dfrac{\mathsf{d}_0, \dots, \mathsf{d}_q, \mathsf{b}_0, \dots, \mathsf{b}_m, \Delta \vdash \mathsf{term} : C}{\mathsf{b}_0, \dots, \mathsf{b}_m, (\mathsf{c}_0, \dots, \mathsf{c}_p \gg_\rho \Delta) \vdash \mathsf{g}(\mathsf{c}_0, \dots, \mathsf{c}_p) \to (\mathsf{d}_0, \dots, \mathsf{d}_q) \fatsemi \quad \mathsf{term} : C}}{\mathsf{a}_0, \dots, \mathsf{a}_n \gg_\sigma (\mathsf{c}_0, \dots, \mathsf{c}_p \gg_\rho \Delta) \vdash \begin{array}{l} \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to (\mathsf{b}_0, \dots, \mathsf{b}_m) \fatsemi \\ \quad \mathsf{g}(\mathsf{c}_0, \dots, \mathsf{c}_p) \to (\mathsf{d}_0, \dots, \mathsf{d}_q) \fatsemi \\ \quad \mathsf{term} : C \end{array}}$$

We now argue for this. In the second line of the derivation we use that, because of variables being distinct, the only possible context $(\mathsf{c}_0, \dots, \mathsf{c}_p) \gg_\tau \Psi$ we can use must factor as $\mathsf{b}_0, \dots, \mathsf{b}_m, (\mathsf{c}_0, \dots, \mathsf{c}_p) \gg_\rho \Delta$ for some Δ and ρ. We can then check

by the form of the rules that the final context Γ must be of the form $\mathsf{a}_0, \dots, \mathsf{a}_n \gg_\sigma (\mathsf{c}_0, \dots, \mathsf{c}_p \gg_\rho \Delta)$ for some insertion σ. All this means that the following derivation is also valid.

$$\dfrac{\dfrac{\mathsf{b}_0, \dots, \mathsf{b}_m, \mathsf{d}_0, \dots, \mathsf{d}_q, \Delta \vdash \mathsf{term} : X}{\mathsf{d}_0, \dots, \mathsf{d}_q, (\mathsf{c}_0, \dots, \mathsf{c}_p \gg \Delta) \vdash \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to \mathsf{b}_0, \dots, \mathsf{b}_m \fatsemi t : X}}{\mathsf{a}_0, \dots, \mathsf{a}_n \gg (\mathsf{c}_0, \dots, \mathsf{c}_p \gg \Delta) \vdash \mathsf{g}(\mathsf{c}_0, \dots, \mathsf{c}_p) \to (\mathsf{d}_0, \dots, \mathsf{d}_q) \fatsemi \mathsf{f}(\mathsf{a}_0, \dots, \mathsf{a}_n) \to (\mathsf{b}_0, \dots, \mathsf{b}_m) \fatsemi \mathsf{term} : X}$$

This derivation proves that the right side term is also valid. □

Finally, we can start proving that do-notation terms are a syntax for symmetric monoidal categories: they form the free strict symmetric monoidal category over a signature.

Lemma 1.4.20. *Do-notation terms over a polygraph $\mathcal{G}$, quotiented by the interchange rule, define a monoidal category,* $\mathsf{Do}(\mathcal{G})$. *This construction induces a functor* $\mathsf{Do} \colon \mathbf{PolyGraph} \to \mathbf{SymMonCat}_{\mathsf{Str}}$.

Proof sketch. We will describe the operations of this strict symmetric monoidal category. Let us start by **composition**: consider derivations $\Gamma \vdash \mathsf{t} \colon B_1 \otimes \dots \otimes B_m$ and $\mathsf{b}_1{:}B_1, \dots, \mathsf{b}_m{:}B_m \vdash \mathsf{s} \colon \Psi$. We will construct, by induction over t, a derivation $\Gamma \vdash \mathsf{comp}(t, s) \colon \Psi$, that will define their *composition*. If t is a return statement, then Γ is a permutation of $\mathsf{b}_1{:}B_1, \dots, \mathsf{b}_m{:}B_m$ and, by exchange, we obtain $\Gamma \vdash s \colon \Psi$ and we define it as the composition. Whenever t consists of a generator followed by a derivation, $\Gamma \vdash t \colon \Psi$ must be of the form

$$(\mathsf{x}_1, \dots, \mathsf{x}_n) \gg \Gamma' \vdash \mathsf{f}(\mathsf{x}_1, \dots, \mathsf{x}_n) \to \mathsf{y}_1, \dots, \mathsf{y}_m \fatsemi \mathsf{t}' : \Psi,$$

and we define $\mathsf{comp}(t, s)$ to mean $\mathsf{f}(\mathsf{x}_1, \dots, \mathsf{x}_n) \to \mathsf{y}_1, \dots, \mathsf{y}_m \fatsemi \mathsf{comp}(\mathsf{t}', \mathsf{s})$, using the $y_1, \dots, y_m, \Gamma' \vdash t' \colon B_1 \otimes \dots \otimes B_m$ we just obtained. Put simply, we have removed the last return statement from one of the derivations, taking care of permutations, and then concatenated both (see Figure 15).

Let us now define **tensoring**: consider derivations $\Gamma \vdash t \colon \Delta$ and $\Gamma' \vdash t' \colon \Delta'$. We start by noticing that, if we have a derivation $\Gamma \vdash \mathsf{t} \colon \Delta$, we can construct, by induction, a derivation $\Gamma, \mathsf{z}{:}Z \vdash \mathsf{t}_z \colon \Delta \otimes Z$ for any $\mathsf{z}{:}Z$: the return case consists of adding an extra variable to the permutation, the generator case is not affected by the presence of an extra variable. Repeating this reasoning and using exchange, we can obtain terms $\Gamma, \Gamma' \vdash \mathsf{t}_{\Gamma'} \colon \Delta \otimes \Gamma'$ and $\Delta, \Gamma' \vdash \mathsf{t'}_{\Delta} \colon \Delta \otimes \Delta'$ that we can compose into $\Gamma, \Gamma' \vdash \mathsf{comp}(\mathsf{t}_{\Gamma'}, \mathsf{t'}_{\Delta'}) \colon \Delta \otimes \Delta'$. The order of composition does not matter because of the *interchange law*. Put simply, we write the two terms one after the other, taking care not to mix the variables (see Figure 15, for a graphical intuition).

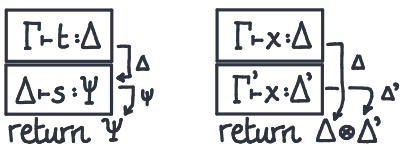


Figure 15. Composition and tensoring in do-notation.

Finally, we must define **identities**, **generators** and **symmetries**. These are the following three terms.

$$\begin{aligned} a_1, \dots, a_n &\vdash \mathsf{return}(a_1, \dots, a_n) \\ a_1, \dots, a_n &\vdash f(a_1, \dots, a_n) \to (b_1, \dots, b_m) \fatsemi \mathsf{return}(b_1, \dots, b_m) \\ a_1, \dots, a_n, b_1, \dots b_m &\vdash \mathsf{return}(b_1, \dots, b_m, a_1, \dots, a_n) \end{aligned}$$

We can check that these operations define a category and a monoidal category. The interchange law of monoidal categories is extracted from the interchange law that we imposed in do-notation. □

THEOREM 1.4.21. *There is an adjunction between polygraphs and the category of strict symmetric monoidal categories given by do-notation terms,* $\mathsf{Do} \dashv \mathsf{Forget}$. *Moreover, do-notation terms and string diagrams are naturally isomorphic.*

PROOF SKETCH. We have already proven that do-notation constructs a symmetric monoidal category over a polygraph. We need to show that there is a unique map out of the category constructed by do-notation that commutes with any assignment of the polygraph to a monoidal category.

The first part would be to prove the initiality of the syntax. We can do so by structural induction: any term is either a return statement or a composition of a generator with a symmetry and a term. In the first case, the symmetry determined by the return statement must be mapped to the corresponding symmetry in any symmetric monoidal category; in the second case, the image of the generator is determined, and the rest of the term is determined by structural induction. In conclusion, the image of any do-notation term is determined in any symmetric monoidal category over which we map the polygraph of generators.

The core of the proof is in showing that this unique possible assignment is well-defined: the only equality imposed on do-notation is the interchange law, but this law corresponds, under the assignment, to the interchange law of symmetric monoidal categories.

Finally, both functors,

$$\mathsf{Do}, \mathsf{String}_\sigma : \mathbf{PolyGraph} \to \mathbf{SymMonCat}_{\mathsf{Str}}$$

have been found to be left adjoints to the same forgetful functor. This implies they are isomorphic functors. □

1.4.6. Example: the XOR Variable Swap. Let us provide an example of the formal usage of both do-notation and string diagrams, by reasoning about a simple process. In imperative programming languages, swapping the value of two variables usually requires a third temporary variable. However, the XOR variable swap algorithm uses the properties of the exclusive-or operation (XOR, $\oplus$) to exchange variables without needing a temporary variable. Let $\mathsf{xor}(x, y) = (x \oplus y, y)$. The code is in Figure 16.

```
xorExchange(x,y):
  xor(x,y) -> (x,y)
  xor(y,x) -> (y,x)
  xor(x,y) -> (x,y)
  return (x,y)
```

FIGURE 16. XOR variable exchange.

The property that makes this algorithm possible is the *nilpotency* of the XOR operation: $x \oplus x = 0$ for any n-bit word $x \in 2^n$. This means that we can prove the correctness of the XOR variable exchange in the abstract setting of nilpotent bialgebras. In fact, consider a polygraph $\mathcal{X}$ with a single object and the generators depicted in Figure 17.

FIGURE 17. Signature for a bialgebra.

We now want to impose a set of equations, $E \subseteq \mathsf{String}(\mathcal{X}) \times \mathsf{String}(\mathcal{X})$, on top of this signature. This can also be done via the adjunction: the equations give two maps $E \to \mathsf{Forget}(\mathsf{String}(\mathcal{X}))$, or equivalently, $\mathsf{String}(E) \to \mathsf{String}(\mathcal{X})$. The coequalizer of the latter two describes the universal monoidal category with some generators and satisfying some equations. Back to our example, the theory of a nilpotent bialgebra satisfies the following equations in Figure 18.

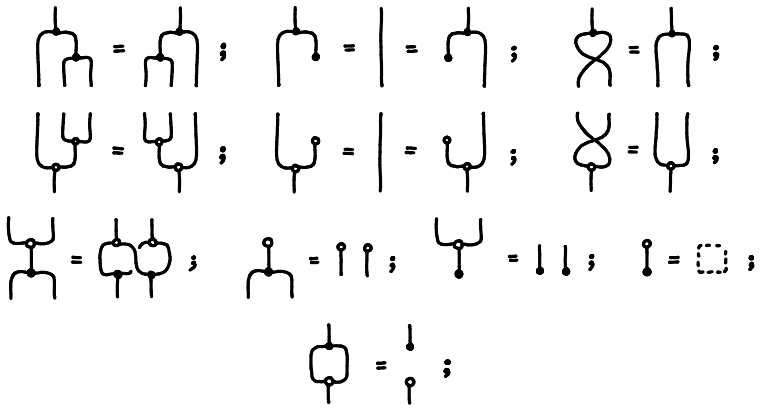


FIGURE 18. Theory of a nilpotent bialgebra.

Given any nilpotent bialgebra in any strict symmetric monoidal category, there exists a unique monoidal functor from the string diagrams quotiented by these equations to that signature.

Proposition 1.4.22. *Let a nilpotent bialgebra in a symmetric monoidal category. The XOR variable exchange algorithm is equal to the swap morphism.*

PROOF. In the theory of nilpotent bialgebras over a symmetric monoidal category, the following equation in Figure 19 holds.

The left hand side represents the XOR variable exchange while the right hand side represents swapping the contents of two variables. We have shown both are equal. □

1.4.7. Bibliography. Symmetric monoidal categories and their hexagon coherence equations were already stated by MacLane [Mac63, ML71], Bénabou defined *commutations* on a monoidal category and an abstract notion of commutative monoid [Bén68]. String diagrams for symmetric monoidal categories are already described by Joyal and Street [JS91, Sel10]; we follow the representation in terms of hypergraphs by Bonchi, Gadducci, Kissinger, Sobocinski and Zanasi [BGK+16].

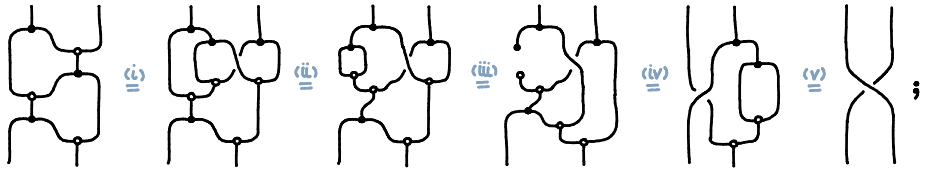


FIGURE 19. XOR variable exchange.

The XOR example was known to Erbele [BE14] and Sobocinski. The idea of using a Rosetta stone to translate between categories and logics comes from Baez and Stay [BS10].

Do-notation comes from the Haskell programming language [HJW+92], where it takes semantics in a strong promonad [Hug00, HJ06] (also known as an *arrow* [Pat03]). We have studied here an adaptation to the monoidal setting. Our presentation of do-notation follows the style of Shulman's categorical logic [Shu16].

1.5. Cartesianity: Determinism and Totality

Our process theories are, by default, *linear on resources*: every variable must be used exactly once. This may seem like a limitation, but it is a more general case that can be particularized into the classical case when necessary: we say that a process theory – a monoidal category – is *classical* or *cartesian* whenever it has processes representing copying and discarding and satisfying suitable axioms.

Non-classical theories can become so in two ways: either because they do not allow copying, or because they do not allow discarding. Theories without copying model stochasticity and non-determinism: running a computation twice is different from just running it once and assuming its result will be the same next time. Theories without discarding model partiality: even if we do not care about the result, we cannot assume anymore that any computation will succed.

1.5.1. Cartesian Monoidal Categories. Cartesian monoidal categories give a universal property to their tensor: the tensor of two objects, $A \times B$, is such that pair of maps to A and B are in precise correspondence to single map to $A \times B$. This universal property, in some sense, ensures that the tensor contains nothing more and nothing less than its two constituent parts; this is what characterizes classical theories.

Definition 1.5.1. *Cartesian monoidal categories* are monoidal categories, $(\mathbb{C}, \times, 1)$, such that

(1) each tensor, $A \times B$, is endowed with projections, $\pi_1 \colon A \times B \to A$ and $\pi_2 \colon A \times B \to B$, that make it a product: for each object $X \in \mathbb{C}_{obj}$ and any pair of morphisms, $f \colon X \to A$ and $g \colon X \to B$, there exists a unique $\langle f, g \rangle \colon X \to A \times B$ such that

$$\langle f, g \rangle \fatsemi \pi_1 = f \quad \text{and} \quad \langle f, g \rangle \fatsemi \pi_2 = g;$$

(2) the unit, 1, is terminal: for each object $X \in \mathbb{C}_{obj}$ there exists a unique morphism $\pi \colon X \to 1$.

Fox's theorem is a characterisation of classical theories, cartesian monoidal categories, in terms of the existence of a uniform cocommutative comonoid structure (copy and delete) on all objects of a monoidal category.

Theorem 1.5.2 (Fox, [Fox76]). *A symmetric monoidal category $(\mathbb{C}, \otimes, I)$ is cartesian monoidal if and only if every object $X \in \mathbb{C}$ has a commutative comonoid structure $(X, \varepsilon_X, \delta_X)$, every morphism of the category $f \colon X \to Y$ is a comonoid homomorphism, and this structure is uniform across the monoidal category, meaning that $\varepsilon_{X \otimes Y} = \varepsilon_X \otimes \varepsilon_Y$, that $\varepsilon_I = \mathrm{id}$, that $\delta_I = \mathrm{id}$ and that $\delta_{X \otimes Y} = (\delta_X \otimes \delta_Y) \fatsemi (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id})$.*

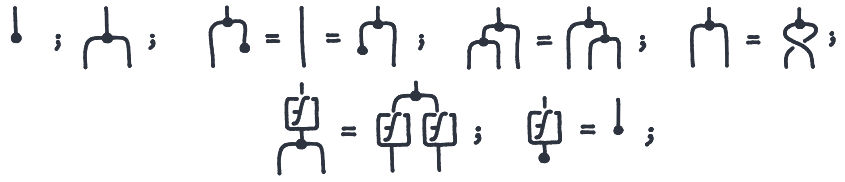


Figure 20. Theory of cartesian categories.

Fox's characterization has a direct translation to string diagrams: the first conditions impose a natural commutative comonoid structure on each generator

(Figure 20); the last conditions state that the structure on all the objects follows from that of the generators.

We can add a slight improvement. Most sources ask the comonoid structure in Fox's theorem (Theorem 1.5.2) to be cocommutative [Fox76, FS19]. However, cocommutativity and coassociativity of the comonoid structure are implied by the fact that the structure is uniform and natural. We present an original refined version of Fox's theorem.

THEOREM 1.5.3 (Refined Fox's theorem). *A symmetric monoidal category, $(\mathbb{C}, \otimes, I)$, is cartesian monoidal if and only if every object $X \in \mathbb{C}$ has a counital comagma structure $(X, \varepsilon_X, \delta_X)$, or $(X, \bullet_X, \curlywedge_X)$, every morphism of the category $f\colon X \to Y$ is a comagma homomorphism, and this structure is uniform across the monoidal category: meaning that $\varepsilon_{X\otimes Y} = \varepsilon_X \otimes \varepsilon_Y$, $\varepsilon_I = \mathrm{id}$, $\delta_I = \mathrm{id}$ and $\delta_{X\otimes Y} = (\delta_X \otimes \delta_Y); (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id})$.*

PROOF. We prove that such a comagma structure is necessarily coassociative and cocommutative. Note that any comagma homomorphism $f\colon A \to B$ must satisfy $\delta_A \fatsemi (f \otimes f) = f \fatsemi \delta_B$. In particular, $\delta_X\colon X \to X \otimes X$ must itself be a comagma homomorphism (see Figure 25), meaning that

$$\delta_X \fatsemi (\delta_X \otimes \delta_X) = \delta_X \fatsemi \delta_{X\otimes X} = \delta_X \fatsemi (\delta_X \otimes \delta_X) \fatsemi (\mathrm{id} \otimes \sigma_{X,Y} \otimes \mathrm{id}), \tag{1}$$

where the second equality follows by uniformity.

FIGURE 21. Comultiplication is a comagma homomorphism.

Now, we prove cocommutativity (Figure 22): composing both sides of Equation (1) with $(\varepsilon_X \otimes \mathrm{id} \otimes \mathrm{id} \otimes \varepsilon_X)$ discards the two external outputs and gives $\delta_X = \delta_X \fatsemi \sigma_X$. Then, we prove coassociativity (Figure 23): composing both sides of

FIGURE 22. Cocommutativity

Equation (1) with $(\mathrm{id} \otimes \varepsilon_X \otimes \mathrm{id} \otimes \mathrm{id})$ discards one of the middle outputs and gives $\delta_X \fatsemi (\mathrm{id} \otimes \delta_X) = \delta_X \fatsemi (\delta_X \otimes \mathrm{id})$.

A coassociative and cocommutative comagma is a cocommutative comonoid. We can then apply the classical form of Fox's theorem (Theorem 1.5.2). □

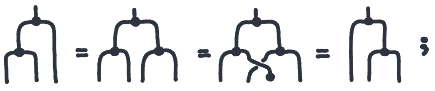


FIGURE 23. Coassociativity

1.5.2. Partial Markov Categories. If process theories were all cartesian, we could use the commutative comonoid structure on every object to simplify calculations. However, most of the theories that pose a challenge to computer science (like stochastic processes, partial processes, or quantum maps) are not cartesian. The rest of this thesis will not assume cartesianity: let us give a good example and motivation for doing this.

Cartesianity in a category with copy and discard processes can be divided in two concepts: *determinism* and *totality*. All pure functions, for instance, are deterministic and total, but stochastic functions are not deterministic (tossing a coin twice is different from tossing it once and copying the result twice), and partially computable functions are not total (because even if we do not care about the output, they could diverge and make the whole process wait).

Definition 1.5.4. In a symmetric monoidal category with uniform commutative comonoids, $(\mathbb{C}, \otimes, I, \copy, \discard, \sigma)$, a morphism $f\colon X \to Y$ is *deterministic* if it can be copied, $f \fatsemi \copy_Y = \copy_X \fatsemi (f \otimes f)$, and it is *total* (or *causal*) if it can be discarded, $f \fatsemi \discard_Y = \discard_X$. Moreover, we say it is *quasitotal* (or *quasicausal*), if it can be copied on the side and discarded, $f = \copy_X \fatsemi ((f \fatsemi \discard_Y) \otimes f)$. See Figure 24.

FIGURE 24. Deterministic, total, and quasitotal morphisms.

Let us provide a single theory where these three assumptions fail: the theory of discrete partial Markov categories [LR23], which we will use to study partial stochastic functions. Apart from copy $(\copy)\colon X \to X \otimes X$ and discard $(\discard)\colon X \to I$, we also consider a comparator morphism $(\curlyvee)\colon X \otimes X \to X$. Copying, discarding and comparing interact as a *partial Frobenius algebra* [DLLNS21].

Definition 1.5.5. A *discrete partial Markov category* is is a symmetric monoidal category $(\mathbb{C}, \otimes, I)$ such that every object has a partial Frobenius monoid structure $(\copy, \discard, \curlyvee)$ that satisfies the axioms in Figure 25 and uniformity, meaning that

(1) $(\copy)_{X\otimes Y} = (\copy_X \otimes \copy_Y) \fatsemi (\mathrm{id}_X \otimes \sigma \otimes \mathrm{id}_Y)$ and $(\copy)_I = \mathrm{id}$;
(2) $(\curlyvee)_{X\otimes Y} = (\mathrm{id}_X \otimes \sigma \otimes \mathrm{id}_Y) \fatsemi (\curlyvee_X \otimes \curlyvee_Y)$ and $(\curlyvee)_I = \mathrm{id}$;
(3) $(\discard)_{X\otimes Y} = (\discard_X \otimes \discard_Y)$ and $(\discard)_I = \mathrm{id}$.

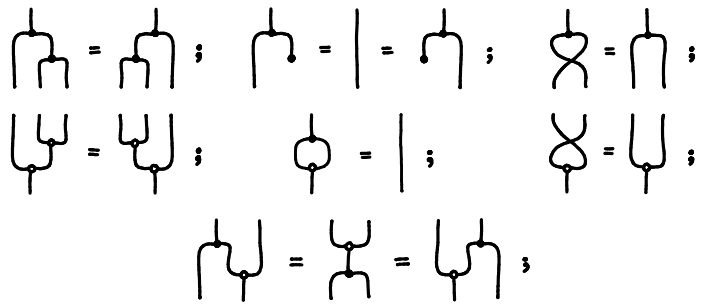


FIGURE 25. Theory of a partial Frobenius algebra.

Proposition 1.5.6. *A subdistribution on a set X is a function $d\colon X \to [0,1]$ that is non-zero on a finite number of elements and that adds up to less or equal than one,*

$$\sum_{d(x)>0} d(x) \leq 1.$$

Subdistributions form a monad, and the Kleisi category of the subdistribution monad forms a discrete partial Markov category [LR23].

Proof. Let us write $\mathbf{D}_{\leq 1}(X)$ for the set of subdistributions over a set. We claim that this extends to a monad in the category of sets and functions $\mathbf{D}_{\leq 1}\colon \mathbf{Set} \to \mathbf{Set}$. The multiplication $\mu\colon \mathbf{D}_{\leq 1}(\mathbf{D}_{\leq 1}(X)) \to \mathbf{D}_{\leq 1}(X)$ is defined by $\mu(\psi)(x) = \sum_{\psi(d)>0} d(x)$, and the unit $\eta\colon X \to \mathbf{D}_{\leq 1}(X)$ is defined by the *Dirac's delta*, $\eta(x_0)(x) = [x = x_0]$, which is valued to 1 whenever $x = x_0$ and is valued to 0 otherwise.

The Kleisli category of this monad has morphisms the *partial stochastic channels* $f\colon X \to \mathbf{D}_{\leq 1}(Y)$. We write $f(y|x) \in [0,1]$ for the value of $f(x)$ on the input $y \in Y$, capturing the usual notation for conditionals in probability. Under this notation, composition on the Kleisli category becomes

$$(f \fatsemi g)(z|x) = \sum_{y \in Y} f(y|x) \cdot g(z|y).$$

While tensoring is $(f \otimes f')(y, y'|x, x') = f(y|x) \cdot f'(y'|x')$. The copy morphism is defined by $(⅄)(x, y|z) = [x = y = z]$; the discard morphism is defined by $(⅂)(|x) = 1$; and the comparator is given by $(⅄)(x|y, z) = [x = y = z]$. It is direct to check that these satisfy the axioms of a partial Frobenius algebra. □

Remark 1.5.7 (Effect algebras)**.** The set of partial stochastic channels between two sets, $X \to \mathbf{D}_{\leq 1}(Y)$, forms a particular algebraic structure known as an *effect module*.

An *effect algebra* [FB94, Jac15, vdW21] is a set E with a partial binary operation $(\oplus)\colon E \times E \to E$, a unary operation $(\bullet)^\perp\colon E \to E$, and a constant $0 \in E$. We write $x \perp y$ whenever $x \oplus y$ is well-defined and we write $1 = 0^\perp$. The effect algebra must satisfy

(1) Commutativity, $x \oplus y = y \oplus x$, where $x \perp y$ implies $y \perp x$.
(2) Unitality, $x \oplus 0 = x = 0 \oplus x$, where $x \perp 0$.
(3) Associativity, $x \oplus (y \oplus z) = (x \oplus y) \oplus z$, where having both $y \perp z$ and $x \perp (y \oplus z)$ implies both $x \perp y$ and $(x \oplus y) \perp z$.
(4) Complementarity, $x^\perp$ is unique such that $x^\perp \oplus x = 1$ and $x \perp x^\perp$.

An effect algebra homomorphism $f\colon E \to F$ must satisfy $f(1) = 1$, and $x \perp y$ must imply $f(x) \perp f(y)$, with $f(x \oplus y) = f(x) \oplus f(y)$. Effect algebras form a symmetric monoidal category.

The unit interval, $[0,1]$, forms an effect algebra with the binary sum (whenever it is contained on the interval), the unary complement $x^\perp = 1 - x$, and the zero; moreover, the unit interval $[0,1]$ with multiplication forms a monoid in the monoidal category of effect algebras.

Partial stochastic channels also form an effect algebra, with the same pointwise operations, but moreover, we can multiply them by a 'scalar' from the unit interval $[0,1]$: they form a module in the monoidal category of effect algebras. The structure of effect module interplays well with composition and tensoring of partial stochastic channels; so we will employ it later in Section 4.5.5.

Remark 1.5.8 (Bayesian inversions)**.** The *Bayesian inversion* of a stochastic channel $g\colon X \to Y$ with respect to a distribution $f\colon I \to X$ is the stochastic channel

$g^\dagger_f \colon Y \to X$ classically defined by

$$g^\dagger_f(x|y) = \frac{g(y|x) \cdot f(x)}{\sum_{x_\bullet \in X} g(y|x_\bullet) \cdot f(x_\bullet)}.$$

Bayesian inversions can be defined synthetically in any partial Markov category [CJ19]. The Bayesian inversion of a morphism $g \colon X \to Y$ with respect to $f \colon I \to X$ is a morphism $g^\dagger_f \colon Y \to X$ satisfying the equation in Figure 26, which translates to the above when the partial Markov category is that of subdistributions.

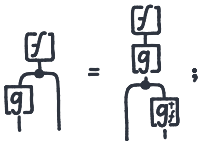


FIGURE 26. Bayesian inversion.

The definition of a *partial Markov category* includes asking a quasitotal Bayesian inversion for every morphism, with respect to any other morphism: this is the notion of *quasitotal conditional* [LR23], which we will not need in this thesis.

Finally, let us justify that this process theory is enough to capture many of the features of classical probability theory. We will prove a synthetic version of Bayes theorem using the syntax of discrete partial Markov categories. The classical Bayes theorem prescribes that, after observing the output of a prior distribution through a channel, we should update our posterior distribution to be the Bayesian inversion of the channel with respect to the prior distribution, evaluated on the observation.

THEOREM 1.5.9 (Synthetic Bayes' Theorem). *In a discrete partial Markov category, observing a deterministic $x \colon I \to X$ from a prior distribution $f \colon I \to A$ through a channel $g \colon A \to X$ is the same, up to scalar, as the Bayesian inversion evaluated on the observation, $g^\dagger_f(x) \colon I \to A$.*

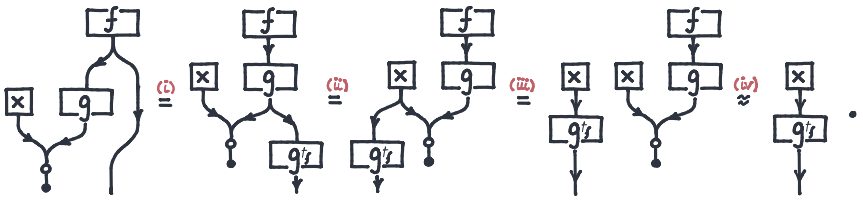


FIGURE 27. Bayes theorem.

PROOF. We employ string diagrams (Figure 27). Equalities follow from *(i)* the definition of Bayesian inversion, *(ii)* the partial Frobenius axioms, and *(iii)* determinism of y. □

1.5.3. Bibliography. Fox's theorem, in its original formulation, is the construction of a right adjoint to the forgetful functor from cartesian monoidal categories to Monoidal categories. This right adjoint is given by the category of cocommutative comonoids over a monoidal category [Fox76]. The version here presented is esentially equivalent; in fact, it is called "Fox's theorem" in the work of Bonchi, Seeber and Sobocinski [BSS18].

The categorical approach to probability theory based on Markov categories is due to Fritz [Fri20] and prior work of Cho and Jacobs [CJ19]. Multiple results of classical probability theory have been adapted to the framework of Markov categories by multiple authors [FR20, FPR21, FP19, FGP21]. Markov categories have been further applied for formalising Bayes networks and other kinds of probabilistic and causal reasoning in categorical terms [Fon13, JZ20, JKZ21]. Their partial counterpart and the application to decision theory was introduced in joint work with Di Lavore [LR23]. Effect algebras are due to Foulis and Bennett [FB94]; Jacobs employed them for a probabilistic categorical logic [Jac15]; van de Wetering [vdW21] characterized the unit interval in terms of effect algebras.

1.6. Premonoidal Categories

It might seem that monoidal categories are limited to pure imperative programming without effects. After all, the *interchange law* seems to imply that the order in which two independent are executed does not matter. This is true, but again, category theory has a solution for us: premonoidal categories.

Category theory has two successful applications that are rarely combined: monoidal string diagrams [JS91] and programming semantics [Mog91]. We use string diagrams to talk about quantum transformations [AC09], relational queries [BSS18], and even computability [Pav13]; at the same time, proof nets and the geometry of interaction [Gir89, BCST96] have been widely applied in computer science [AHS02, HMH14]. On the other hand, we traditionally use monads and comonads, Kleisli categories and premonoidal categories to explain effectful functional programming [Hug00, JHH09, Mog91, PT99, UV08]. Even if we traditionally employ Freyd categories with a cartesian base [Pow02], we can also consider non-cartesian Freyd categories [SL13], which we call *effectful categories*.

This section introduces premonoidal categories and effectful categories. The next section will study their string diagrams in terms of monoidal categories, reducing them to a particular consideration on top of the theory of monoidal categories.

1.6.1. Premonoidal Categories. Premonoidal categories are monoidal categories without the *interchange law*, $(f \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes g) \neq (\mathrm{id} \otimes g) \fatsemi (f \otimes \mathrm{id})$. This means that we cannot tensor any two arbitrary morphisms, $(f \otimes g)$, without explicitly stating which one is to be composed first, $(f \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes g)$ or $(\mathrm{id} \otimes g) \fatsemi (f \otimes \mathrm{id})$, and the two compositions are not equivalent (Figure 28).

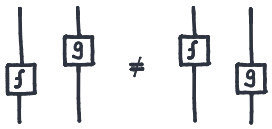


FIGURE 28. The interchange law does not hold in a premonoidal category.

In technical terms, the tensor of a premonoidal category $(\otimes) \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$ is not a functor, but only what is called a *sesquifunctor*: independently functorial in each variable. Tensoring with any identity is itself a functor $(\bullet \otimes \mathrm{id}) \colon \mathbb{C} \to \mathbb{C}$, but there is no functor $(\bullet \otimes \bullet) \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$.

A practical motivation for dropping the interchange law can be found when describing transformations that affect a global state. These effectful processes should not interchange in general, because the order in which we modify the global state is meaningful. For instance, in the Kleisli category of the *writer monad*, $(\Sigma^* \times \bullet) \colon \mathbf{Set} \to \mathbf{Set}$ for some alphabet $\Sigma \in \mathbf{Set}$, we can consider the function $\mathsf{print} \colon \Sigma^* \to \Sigma^* \times 1$. The order in which we "print" does matter (Figure 29).

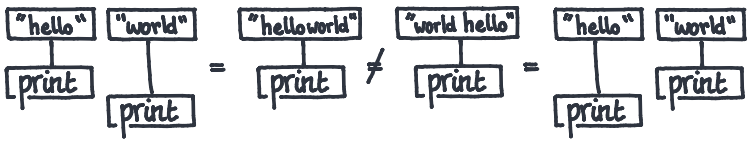


FIGURE 29. Writing does not interchange.

Not surprisingly, the paradigmatic examples of premonoidal categories are the Kleisli categories of Set-based monads $T \colon \mathbf{Set} \to \mathbf{Set}$ (more generally, of strong

monads), which fail to be monoidal unless the monad itself is commutative [Gui80, PR97, PT99, Hed19]. Intuitively, the morphisms are "effectful", and these effects do not always commute.

However, we may still want to allow some morphisms to interchange. For instance, apart from asking the same associators and unitors of monoidal categories to exist, we ask them to be *central*: which means that they interchange with any other morphism. This notion of centrality forces us to write the definition of premonoidal category in two different steps: first, we introduce the minimal setting in which centrality can be considered (*binoidal categories* [PT99]) and then we use that setting to bootstrap the full definition of premonoidal category with central coherence morphisms.

Definition 1.6.1 (Binoidal category)**.** A *binoidal category* is a category $\mathbb{C}$ endowed with an object $I \in \mathbb{C}$ and an object $A \otimes B$ for each $A \in \mathbb{C}$ and $B \in \mathbb{C}$. There are functors $(A \otimes \bullet) \colon \mathbb{C} \to \mathbb{C}$, and $(\bullet \otimes B) \colon \mathbb{C} \to \mathbb{C}$ that coincide on $(A \otimes B)$. Note that $(\bullet \otimes \bullet)$ is not being defined as a functor.

Again, this means that we can tensor with identities (whiskering), functorially; but we cannot tensor two arbitrary morphisms: the interchange law stops being true in general. The *centre*, $\mathcal{Z}(\mathbb{C})$, is the wide subcategory of morphisms that do satisfy the interchange law with any other morphism. That is, $f \colon A \to B$ is *central* if, for each $g \colon A' \to B'$,

$$(f \otimes \mathrm{id}_{A'}) \fatsemi (\mathrm{id}_B \otimes g) = (\mathrm{id}_A \otimes g) \fatsemi (f \otimes \mathrm{id}_{B'}), \text{ and}$$
$$(\mathrm{id}_{A'} \otimes f) \fatsemi (g \otimes \mathrm{id}_B) = (g \otimes \mathrm{id}_A) \fatsemi (\mathrm{id}_{B'} \otimes f).$$

Definition 1.6.2. A *premonoidal category* is a noidal category $(\mathbb{C}, \otimes, I)$ together with the following coherence isomorphisms $\alpha_{A,B,C} \colon A \otimes (B \otimes C) \to (A \otimes B) \otimes C$, $\rho_A \colon A \otimes I \to A$ and $\lambda_A \colon I \otimes A \to A$ which are central, natural *separately at each given component*, and satisfy the pentagon and triangle equations.

A premonoidal category is *strict* when these coherence morphisms are identities. A premonoidal category is moreover *symmetric* when it is endowed with a coherence isomorphism $\sigma_{A,B} \colon A \otimes B \to B \otimes A$ that is central and natural at each given component and satisfies the symmetry condition and hexagon equations.

Remark 1.6.3. The coherence theorem of monoidal categories still holds for premonoidal categories: every premonoidal is equivalent to a strict one. We will construct the free strict premonoidal category using string diagrams. However, the usual string diagrams for monoidal categories need to be restricted: in premonoidal categories, we cannot consider two morphisms in parallel unless any of the two is *central*.

1.6.2. Effectful and Freyd Categories. Premonoidal categories immediately present a problem: what are the premonoidal functors? If we want them to compose, they should preserve the centrality of the coherence morphisms (so that the central coherence morphisms of $F \fatsemi G$ are these of F after applying G), but naively asking them to preserve all central morphisms rules out important examples [SL13]. The solution is to explicitly choose some central morphisms that represent "pure" computations. These do not need to form the whole centre: it could be that some morphisms considered *effectful* just "happen" to fall in the centre of the category, while we do not ask our functors to preserve them. This is the well-studied notion of a *non-cartesian Freyd category*, which we shorten to *effectful monoidal category* or *effectful category*.

Effectful categories are premonoidal categories endowed with a chosen family of central morphisms. These central morphisms are called pure morphisms, contrasting with the general, non-central, morphisms that fall outside this family, which we call effectful.

Definition 1.6.4. An *effectful category* is an identity-on-objects functor $\mathbb{V} \to \mathbb{C}$ from a monoidal category $\mathbb{V}$ (the pure morphisms, or "values") to a premonoidal category $\mathbb{C}$ (the effectful morphisms, or "computations"), that strictly preserves all of the premonoidal structure and whose image is central. It is *strict* when both are. A *Freyd category* [Lev22] is an effectful category where the pure morphisms form a cartesian monoidal category.

Effectful categories solve the problem of defining premonoidal functors: a functor between effectful categories needs to preserve only the pure morphisms. We are not losing expressivity: premonoidal categories are effectful with their centre, $\mathcal{Z}(\mathbb{C}) \to \mathbb{C}$. From now on, we study effectful categories.

Definition 1.6.5 (Effectful functor)**.** Let $\mathbb{V} \to \mathbb{C}$ and $\mathbb{W} \to \mathbb{D}$ be effectful categories. An *effectful functor* is a quadruple $(F, F_0, \varepsilon, \mu)$ consisting of a functor $F\colon \mathbb{C} \to \mathbb{D}$ and a functor $F_0\colon \mathbb{V} \to \mathbb{W}$ making the square commute, and two natural and pure isomorphisms $\varepsilon\colon I' \cong F(I)$ and $\mu\colon F(A \otimes B) \cong F(A) \otimes F(B)$ such that they make F_0 a monoidal functor. It is *strict* if these are identities.

When drawing string diagrams in an effectful category, we shall use two different colours to declare if we are depicting either a value or a computation (Figure 30).

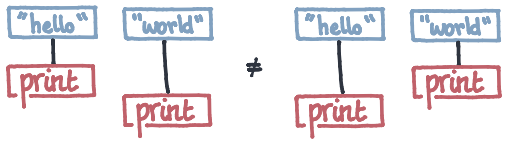


FIGURE 30. "Hello world" is not "world hello".

Here, the values "hello" and "world" satisfy the interchange law as in an ordinary monoidal category. However, the effectful computation "print" does not need to satisfy the interchange law. String diagrams like these can be found in the work of Alan Jeffrey [Jef97b]. Jeffrey presents a clever mechanism to graphically depict the failure of interchange: all effectful morphisms need to have a control wire as an input and output. This control wire needs to be passed around to all the computations in order, and it prevents them from interchanging.

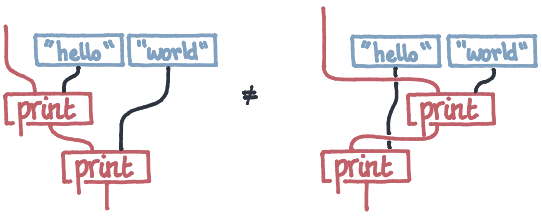


FIGURE 31. An extra wire prevents interchange.

Our interpretation of monoidal categories is as theories of resources. We can interpret premonoidal categories as monoidal categories with an extra resource – the "runtime" – that needs to be passed to all computations. The next section promotes Jeffrey's observation into a theorem.

Remark 1.6.6. After the next section, which reduces premonoidal categories to monoidal categories, the rest of this thesis deals mostly with monoidal categories. Why not study premonoidal categories instead of just monoidal categories? Premonoidal categories should be more general: they do not assume the interchange law, which is false in stateful systems. However, we give an argument for studying only monoidal categories in the next section: any premonoidal category can be reinterpreted as a monoidal category carrying an extra resource (an extra wire) representing the global state. We do not need to write a new theory of premonoidal categories: premonoidal categories are already monoidal categories, in which one wire is hidden. The following section makes this idea formal.

1.6.3. Bibliography. This chapter follows closely the first part of "Promonads and String Diagrams for Effectful Categories", by this author [Rom22].

Effectful categories are the monoidal counterpart of a well-known notion: that of "Freyd categories". The name "Freyd category" sometimes assumes cartesianity of the pure morphisms, but it is also used for the general case; choosing to call "effectful categories" to the general case and reserving the name "Freyd categories" for the cartesian ones avoids this clash of nomenclature. There exists also the more fine-grained notion of "Cartesian effect category" [DDR11], which generalizes Freyd categories and may further justify calling "effectful category" to the general case.

1.7. String Diagrams for Premonoidal Categories

Premonoidal categories give us a generalization of monoidal categories accounting for effectful computation but, at the same time, they do not need us to change our syntax yet. String diagrams for premonoidal categories can be reduced to string diagrams for monoidal categories: they rely on the fact that the morphisms of the monoidal category freely generated over a polygraph of generators are string diagrams on these generators, quotiented by topological deformations, as we saw in Section 1.1.3.

1.7.1. Effectful Polygraphs. We justify string diagrams for premonoidal categories by proving that the freely generated effectful category over a pair of polygraphs (for pure and effectful generators, respectively) can be constructed as the freely generated monoidal category over a particular polygraph that includes an extra wire. In the same sense that polygraphs are signatures for generating free monoidal categories, effectful polygraphs are signatures for generating free effectful categories.

Definition 1.7.1. An *effectful polygraph* is a pair of polygraphs $(\mathcal{V}, \mathcal{G})$ sharing the same objects, $\mathcal{V}_{\text{obj}} = \mathcal{G}_{\text{obj}}$. A morphism of effectful polygraphs $(u, f) \colon (\mathcal{V}, \mathcal{G}) \to (\mathcal{W}, \mathcal{H})$ is a pair of morphisms of polygraphs, $u \colon \mathcal{V} \to \mathcal{W}$ and $f \colon \mathcal{G} \to \mathcal{H}$, such that they coincide on objects, $f_{\text{obj}} = u_{\text{obj}}$.

1.7.2. Adding Runtime. Recall from Section 1.1.3 that there exists an adjunction between polygraphs and strict monoidal categories. Any monoidal category $\mathbb{C}$ can be seen as a polygraph, $\mathsf{Forget}(\mathbb{C})$, where the edges are morphisms

$$\mathsf{Forget}(\mathbb{C})(A_0, \dots, A_n; B_0, \dots, B_m) = \mathbb{C}(A_0 \otimes \dots \otimes A_n; B_0 \otimes \dots \otimes B_m),$$

and we forget about composition and tensoring. Given a polygraph $\mathcal{G}$, the free strict monoidal category, which we will now write as $\mathrm{Mon}(\mathcal{G}) = \mathsf{String}(\mathcal{G})$, is the strict monoidal category that has as morphisms the string diagrams over the generators of the polygraph.

We will construct a similar adjunction between effectful polygraphs and effectful categories. Let us start by formally adding the runtime to a free monoidal category.

Definition 1.7.2 (Runtime monoidal category)**.** Let $(\mathcal{V}, \mathcal{G})$ be an effectful polygraph. Its *runtime monoidal category*, $\mathrm{Mon}_{\mathrm{Run}}(\mathcal{V}, \mathcal{G})$, is the monoidal category freely generated from adding an extra object – the runtime, R – to the input and output of every effectful generator in $\mathcal{G}$ (but not to those in $\mathcal{V}$), and letting that extra object be braided with respect to every other object of the category.

In other words, it is the monoidal category freely generated by the following polygraph, $\mathrm{Run}(\mathcal{V}, \mathcal{G})$, (Figure 32), assuming $A_0, \dots, A_n$ and $B_0, \dots, B_m$ are distinct from R

- $\mathrm{Run}(\mathcal{V}, \mathcal{G})_{\text{obj}} = \mathcal{G}_{\text{obj}} + \{R\} = \mathcal{V}_{\text{obj}} + \{R\}$,
- $\mathrm{Run}(\mathcal{V}, \mathcal{G})(R, A_0, \dots, A_n; R, B_0, \dots, B_n) = \mathcal{G}(A_0, \dots, A_n; B_0, \dots, B_n)$,
- $\mathrm{Run}(\mathcal{V}, \mathcal{G})(A_0, \dots, A_n; B_0, \dots, B_n) = \mathcal{V}(A_0, \dots, A_n; B_0, \dots, B_n)$,
- $\mathrm{Run}(\mathcal{V}, \mathcal{G})(R, A_0; A_0, R) = \mathrm{Run}(\mathcal{V}, \mathcal{G})(A_0, R; R, A_0) = \{\sigma\}$,

with $\mathrm{Run}(\mathcal{V}, \mathcal{G})$ empty in any other case, and quotiented by the braiding axioms for R (Figure 33).

Somehow, we are asking the runtime R to be in the Drinfeld centre [DGNO10] of the monoidal category. The extra wire that R provides is only used to prevent interchange, and so it does not really matter where it is placed in the input and the output. We can choose to always place it on the left, for instance – and indeed we will be able to do so – but a better solution is to just consider objects "up to some runtime braidings". This is formalized by the notion of *braid clique*.

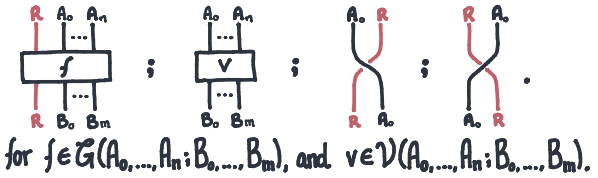


FIGURE 32. Generators for the runtime monoidal category.

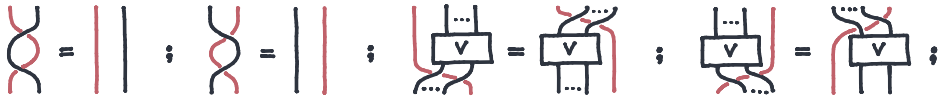


FIGURE 33. Axioms for the runtime monoidal category.

Definition 1.7.3 (Braid clique)**.** Given any list of objects $A_0, \dots, A_n$ in $\mathcal{V}_{\text{obj}} = \mathcal{G}_{\text{obj}}$, we construct a *clique* [Tod10, Shu18] in the category $\text{Mon}_{\text{Run}}(\mathcal{V}, \mathcal{G})$: we consider the objects, $A_0 \otimes \dots \otimes R_{(i)} \otimes \dots \otimes A_n$, created by inserting the runtime R in all of the possible $0 \leq i \leq n+1$ positions; and we consider the family of commuting isomorphisms constructed by braiding the runtime,

$$\sigma_{i,j} \colon A_0 \otimes \dots \otimes R_{(i)} \otimes \dots \otimes A_n \to A_0 \otimes \dots \otimes R_{(j)} \otimes \dots \otimes A_n.$$

We call this the *braid clique*, $\text{Braid}_R(A_0, \dots, A_n)$, on that list.

Definition 1.7.4. A *braid clique morphism*,

$$f \colon \text{Braid}_R(A_0, \dots, A_n) \to \text{Braid}_R(B_0, \dots, B_m),$$

is a family of morphisms in the runtime monoidal category, $\text{Mon}_{\text{Run}}(\mathcal{V}, \mathcal{G})$, from each of the objects of first clique to each of the objects of the second clique,

$$f_{ik} \colon A_0 \otimes \dots \otimes R_{(i)} \otimes \dots \otimes A_n \to B_0 \otimes \dots \otimes R_{(k)} \otimes \dots \otimes B_m,$$

that moreover commutes with all braiding isomorphisms, $f_{ij} \fatsemi \sigma_{jk} = \sigma_{il} \fatsemi f$.

A braid clique morphism $f \colon \text{Braid}_R(A_0, \dots, A_n) \to \text{Braid}_R(B_0, \dots, B_m)$ is fully determined by *any* of its components, by pre/post-composing it with braidings. In particular, a braid clique morphism is always fully determined by its leftmost component $f_{00} \colon R \otimes A_0 \otimes \dots \otimes A_n \to R \otimes B_0 \otimes \dots \otimes B_m$.

Lemma 1.7.5. *Let $(\mathcal{V}, \mathcal{G})$ be an effectful polygraph. There exists a premonoidal category, $\text{Eff}(\mathcal{V}, \mathcal{G})$, that has objects the braid cliques, $\text{Braid}_R(A_0, \dots, A_n)$, in the runtime monoidal category $\text{Mon}_{\text{Run}}(\mathcal{V}, \mathcal{G})$, and as morphisms the braid clique morphisms between them.*

PROOF. First, let us give $\text{Eff}(\mathcal{V}, \mathcal{G})$ the structure of a category. The identity on $\text{Braid}_R(A_0, \dots, A_n)$ is the identity on $R \otimes A$. The composition of a morphism $R \otimes A \to R \otimes B$ with a morphism $R \otimes B \to R \otimes C$ is their plain composition in $\text{Mon}_{\text{Run}}(\mathcal{V}, \mathcal{G})$.

Let us now check that it is moreover a premonoidal category. Tensoring of cliques is given by concatenation of lists, which coincides with the tensor in $\text{Mon}_{\text{Run}}(\mathcal{V}, \mathcal{G})$. However, it is interesting to note that the tensor of morphisms cannot be defined in this way: a morphism $R \otimes A \to R \otimes B$ cannot be tensored with a morphism $R \otimes A' \to R \otimes B'$ to obtain a morphism $R \otimes A \otimes A' \to R \otimes B \otimes B'$.

Whiskering of a morphism $f \colon R \otimes A \to R \otimes B$ is defined with braidings in the left case, $R \otimes C \otimes A \to R \otimes C \otimes B$, and by plain whiskering in the right case, $R \otimes A \otimes C \to R \otimes B \otimes C$, as depicted in Figure 34. Finally, the associators and unitors are identities, which are always natural and central. □

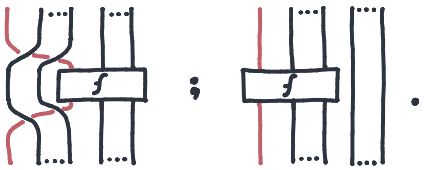


FIGURE 34. Whiskering in the runtime premonoidal category.

Lemma 1.7.6. *Let $(\mathcal{V}, \mathcal{G})$ be an effectful polygraph. There exists an identity-on-objects functor $\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G})$ that strictly preserves the premonoidal structure and whose image is central.*

PROOF. A morphism $v \in \mathrm{Mon}(\mathcal{V})(A, B)$ induces a morphism $(\mathrm{id}_R \otimes v) \in \mathrm{Mon}_{\mathrm{Run}}(\mathcal{V}, \mathcal{G})(R \otimes A, R \otimes B)$, which can be read as a morphism of cliques $(\mathrm{id}_R \otimes v) \in \mathrm{Eff}(\mathcal{V}, \mathcal{G})(A, B)$. This is tensoring with an identity, which is indeed functorial.

Let us now show that this functor strictly preserves the premonoidal structure. The fact that it preserves right whiskerings is immediate. The fact that it preserves left whiskerings follows from the axioms of symmetry (Figure 35, left). Associators and unitors are identities, which are preserved by tensoring with an identity. Finally, we can check by string diagrams that the image of this functor is central,

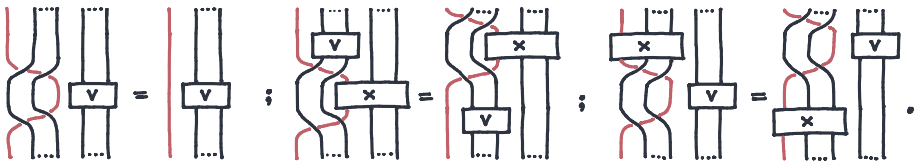


FIGURE 35. Preservation of whiskerings, and centrality.

interchanging with any given $x \colon R \otimes C \to R \otimes D$ (Figure 35, center and right). □

Lemma 1.7.7. *Let $(\mathcal{V}, \mathcal{G})$ be an effectful polygraph and consider the effectful category determined by $\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G})$. Let $\mathbb{V} \to \mathbb{C}$ be a strict effectful category endowed with an effectful polygraph morphism $F \colon (\mathcal{V}, \mathcal{G}) \to \mathcal{U}(\mathbb{V}, \mathbb{C})$. There exists a unique strict effectful functor from $(\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G}))$ to $(\mathbb{V} \to \mathbb{C})$ commuting with F as an effectful polygraph morphism.*

PROOF. By freeness, there already exists a unique strict monoidal functor $H_0 \colon \mathrm{Mon}(\mathcal{V}) \to \mathbb{V}$ that sends any object $A \in \mathcal{V}_{\mathrm{obj}}$ to $F_{obj}(A)$. We will show there is a unique way to extend this functor together with the hypergraph assignment $\mathcal{G} \to \mathbb{C}$ into a functor $H \colon \mathrm{Eff}(\mathcal{V}, \mathcal{G}) \to \mathbb{C}$. Giving such a functor amounts to give some mapping of morphisms containing the runtime R in some position in their input and output,

$$f \colon A_0 \otimes \ldots \otimes R \otimes \ldots \otimes A_n \to B_0 \otimes \ldots \otimes R \otimes \ldots \otimes B_m$$

to morphisms $H(f) \colon FA_0 \otimes \ldots \otimes FA_n \to FB_0 \otimes \ldots \otimes FB_n$ in $\mathbb{C}$, in a way that preserves composition, whiskerings, inclusions from $\mathrm{Mon}(\mathcal{V})$, and that is invariant to composition with braidings. In order to define this mapping, we will perform structural induction over the monoidal terms of the runtime monoidal category of the form $\mathrm{Mon}_{\mathrm{Run}}(\mathcal{V}, \mathcal{G})(A_0 \otimes \ldots \otimes R^{(i)} \otimes \ldots \otimes A_n, R \otimes B_0 \otimes \ldots \otimes R^{(j)} \otimes \ldots \otimes B_m)$ and show that it is the only mapping with these properties (Figure 36).

Monoidal terms in a strict, freely presented, monoidal category are formed by identities (id), composition ($\fatsemi$), tensoring ($\otimes$), and some generators (in this case, in Figure 32). Monoidal terms are subject to *(i)* functoriality of the tensor, $\mathrm{id} \otimes \mathrm{id} = \mathrm{id}$

and $(f \fatsemi g) \otimes (h \fatsemi k) = (f \otimes h) \fatsemi (g \otimes k)$; *(ii)* associativity and unitality of the tensor, $f \otimes \mathrm{id}_I = f$ and $f \otimes (g \otimes h) = (f \otimes g) \otimes h$; *(iii)* the usual unitality, $f \fatsemi \mathrm{id} = f$ and $\mathrm{id} \fatsemi f = f$ and associativity $f \fatsemi (g \fatsemi h) = (f \fatsemi g) \fatsemi h$; *(iv)* the axioms of our presentation (in this case, in Figure 33).

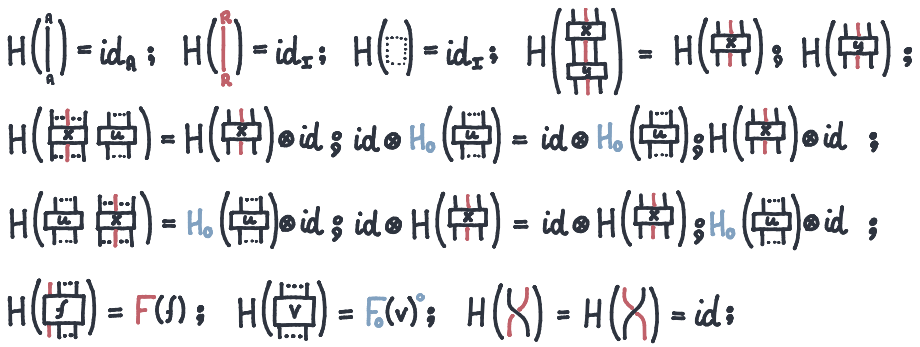


FIGURE 36. Assignment on morphisms, defined by structural induction on terms.

- If the term is an identity, it can be *(i)* an identity on an object $A \in (\mathcal{V}, \mathcal{G})_{\mathrm{obj}}$, in which case it must be mapped to the same identity by functoriality, $H(\mathrm{id}_A) = \mathrm{id}_A$; *(ii)* an identity on the runtime, in which case it must be mapped to the identity on the unit object, $H(\mathrm{id}_R) = \mathrm{id}_I$; or *(iii)* an identity on the unit object, in which case it must be mapped to the identity on the unit, $H(\mathrm{id}_I) = \mathrm{id}_I$.
- If the term is a composition, $(f \fatsemi g) \colon A_0 \otimes \ldots \otimes R \otimes \ldots \otimes A_n \to C_0 \otimes \ldots \otimes R \otimes \ldots \otimes C_k$, it must be along a boundary of the form $B_0 \otimes \ldots \otimes R \otimes \ldots \otimes B_m$: this is because every generator leaves the number of runtimes, R, invariant. Thus, each one of the components determines itself a braid clique morphism. We must preserve composition of braid clique morphisms, so we must map $H(f \fatsemi g) = H(f) \fatsemi H(g)$.
- If the term is a tensor of two terms, $(x \otimes u) \colon A_0 \otimes \ldots \otimes R \otimes \ldots \otimes A_n \to B_0 \otimes \ldots \otimes R \otimes \ldots \otimes B_m$, then only one of them was a term taking R as input and output (without loss of generality, assume it to be the first one) and the other was not: again, by construction, there are no morphisms taking one R as input and producing none, or viceversa. We split this morphism into $x \colon A_0 \otimes \ldots \otimes R \otimes \ldots \otimes A_{i-1} \to B_0 \otimes \ldots \otimes R \otimes \ldots \otimes B_{j-1}$ and $u \colon A_i \otimes \ldots \otimes A_n \to B_j \otimes \ldots \otimes B_m$.

  Again by structural induction, this time over terms $u \colon A_i \otimes \ldots \otimes A_n \to B_j \otimes \ldots \otimes B_m$, we know that the morphism must be either a generator in $\mathcal{V}(A_i, \ldots, A_n; B_j, \ldots, B_n)$ or a composition and tensoring of them. That is, u is a morphism in the image of $\mathrm{Mon}(\mathcal{V})$, and it must be mapped according to the functor $H_0 \colon \mathrm{Mon}(\mathcal{V}) \to \mathbb{V}$.

  By induction hypothesis, we know how to map the morphism $x \colon A_0 \otimes \ldots \otimes R \otimes \ldots \otimes A_{i-1} \to B_0 \otimes \ldots \otimes R \otimes \ldots \otimes B_{j-1}$. This means that, given any tensoring $x \otimes u$, we must map it to $H(x \otimes u) = (H(x) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes H_0(u)) = (\mathrm{id} \otimes H_0(u)) \fatsemi (H(x) \otimes \mathrm{id})$, where $H_0(u)$ is central.
- If the string diagram consists of a single generator, $f \colon R \otimes A \to R \otimes B$, it can only come from a generator $f \in \mathrm{Run}(\mathcal{V}, \mathcal{G})(R, A_0, \ldots, A_n; R, B_0, \ldots, B_m) = \mathcal{G}(A_0, \ldots, A_n; B_0, \ldots, B_m)$, which must be mapped to $H(f) = F(f) \in \mathbb{C}(A_0 \otimes \ldots \otimes A_n, B_0 \otimes \ldots \otimes B_m)$. If the string diagram consists of a single braiding, it must be mapped to the identity, because the want the assignment to be invariant to braidings.

Now, we need to prove that this assignment is well-defined with respect to the axioms of these monoidal terms. Our reasoning follows Figure 37.

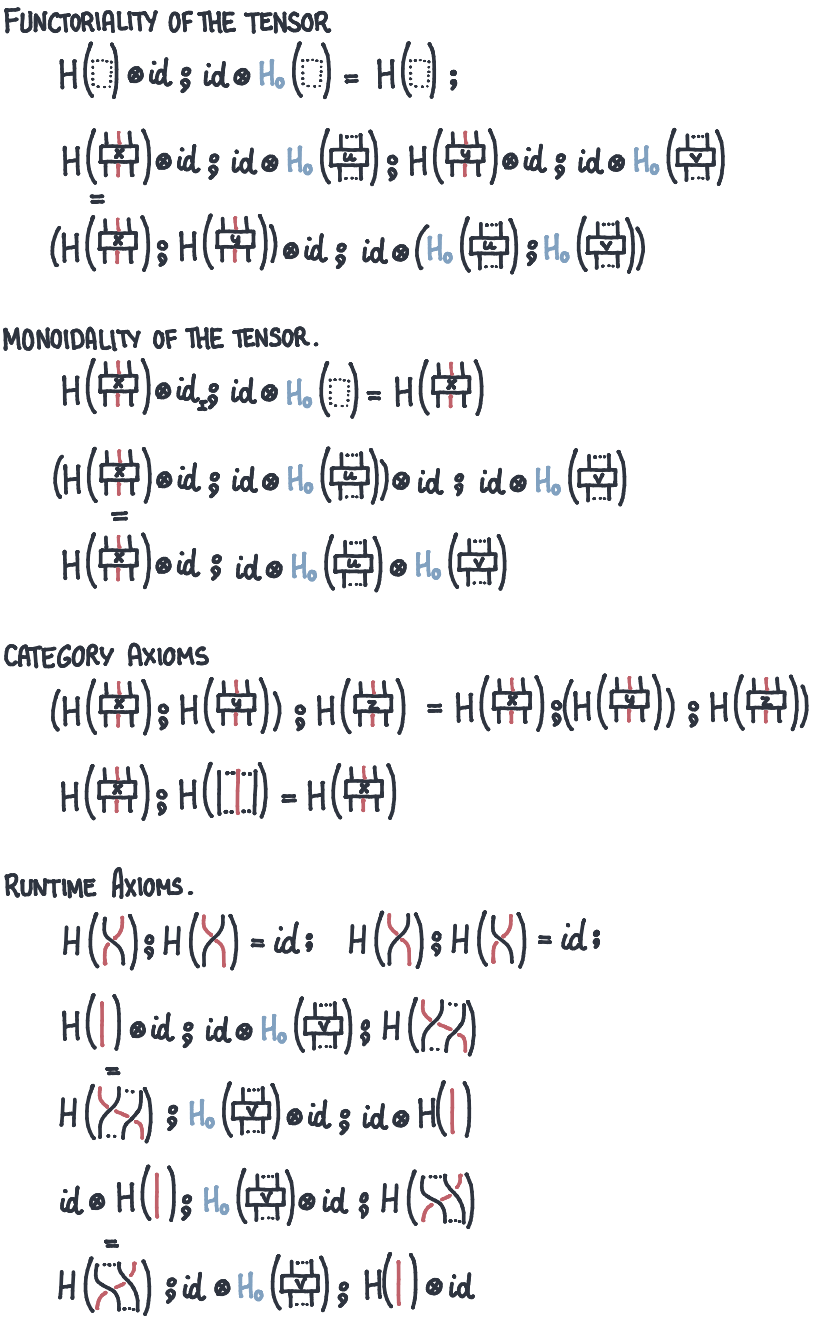


FIGURE 37. The assignment is well defined.

- The tensor is functorial. We know that $H(\mathrm{id} \otimes \mathrm{id}) = H(\mathrm{id})$, both are identities and that can be formally proven by induction on the number of wires. Now, for the interchange law, consider a quartet of morphisms that can be composed or tensored first and such that, without loss of generality, we assume the runtime to be on the left side. Then, we can use centrality to argue that

$$\begin{aligned} H((x \otimes u) \fatsemi (y \otimes v)) &= (H(x) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes H_0(u)) \fatsemi (H(y) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes H_0(v)) \\ &= ((H(x) \fatsemi H(y)) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes (H_0(u) \fatsemi H_0(v))) \end{aligned}$$

$$= H((x \fatsemi y) \otimes (u \fatsemi v)).$$

- The tensor is monoidal. We know that $H(x \otimes \mathrm{id}_I) = (H(x) \otimes \mathrm{id}_I) \fatsemi (\mathrm{id} \otimes \mathrm{id}_I) = H(x)$. Now, for associativity, consider a triple of morphisms that can be tensored in two ways and such that, without loss of generality, we assume the runtime to be on the left side. Then, we can use centrality to argue that

$$\begin{aligned} H((x \otimes u) \otimes v) &= (((H(x) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes H_0(u))) \otimes \mathrm{id}) \fatsemi \mathrm{id} \otimes H_0(v) \\ &= (H(x) \otimes \mathrm{id}) \fatsemi (\mathrm{id} \otimes H_0(u) \otimes H_0(v)) \\ &= H(x \otimes (u \otimes v)) \end{aligned}$$

- The terms form a category. And indeed, it is true by construction that $H(x \fatsemi (y \fatsemi z)) = H((x \fatsemi y) \fatsemi z)$ and also that $H(x \fatsemi \mathrm{id}) = H(x)$ because H preserves composition.
- The runtime category enforces some axioms. The composition of two braidings is mapped to the identity by the fact that H preserves composition and sends both to the identity. Both sides of the braid naturality over a morphism v are mapped to $H_0(v)$; with the multiple braidings being mapped again to the identity.

Thus, H is well-defined and it defines the only possible assignment and the only possible strict premonoidal functor. □

THEOREM 1.7.8 (Runtime as a resource). *The free strict effectful category over an effectful polygraph $(\mathcal{V}, \mathcal{G})$ is $\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G})$. Its morphisms $A \to B$ are in bijection with the morphisms $R \otimes A \to R \otimes B$ of the runtime monoidal category,*

$$\mathrm{Eff}(\mathcal{V}, \mathcal{G})(A, B) \cong \mathrm{Mon}_{\mathrm{Run}}(\mathcal{V}, \mathcal{G})(R \otimes A, R \otimes B).$$

PROOF. We must first show that $\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G})$ is an effectful category. The first step is to see that $\mathrm{Eff}(\mathcal{V}, \mathcal{G})$ forms a premonoidal category (Lemma 1.7.5). We already know that $\mathrm{Mon}(\mathcal{V})$ is a monoidal category: a strict, freely generated one. There exists an identity on objects functor, $\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G})$, that strictly preserves the premonoidal structure and centrality (Lemma 1.7.6).

Let us now show that it is the free one over the effectful polygraph $(\mathcal{V}, \mathcal{G})$. Let $\mathbb{V} \to \mathbb{C}$ be an effectful category, with an effectful polygraph map $F \colon (\mathcal{V}, \mathcal{G}) \to \mathcal{U}(\mathbb{V}, \mathbb{C})$. We can construct a unique effectful functor from $(\mathrm{Mon}(\mathcal{V}) \to \mathrm{Eff}(\mathcal{V}, \mathcal{G}))$ to $(\mathbb{V} \to \mathbb{C})$ giving its universal property (Lemma 1.7.7). □

Corollary 1.7.9 (String diagrams for effectful categories)**.** *We can use string diagrams for effectful categories, quotiented under the same isotopy as for monoidal categories, provided that we do represent the runtime as an extra wire that needs to be the input and output of every effectful morphism.*

1.7.3. Example: a Theory of Global State. Let us provide an example of reasoning using the string diagrams for premonoidal categories. Imperative programs are characterized by the presence of a global state that can be mutated. Reading or writing to this global state constitutes an effectful computation: the order of operations that affect some global state cannot be changed. Let us propose a simple theory of global state and let us show that it is enough to capture the phenomenon of *race conditions*.

Definition 1.7.10. The *theory of global state* is given by a single object X; two pure generators, $(\curlywedge) \colon X \to X \otimes X$ and $(\curlyvee) \colon X \to I$, allowing copy and discard; and two effectful generators, $\mathsf{put} \colon X \to I$ and $\mathsf{get} \colon I \to X$, quotiented by the equations in Figure 39.

These two put and get generators, without extra axioms, are enough for capturing what happens when a process can *send* or *receive* resources; in further sections,

FIGURE 38. Generators of the theory of global state.

we will develop this theory. Right now, we are only concerned with the theory of a single *global state*, accessed by a single process: we can impose some axioms that assert that the memory was not changed by anyone but this single process.

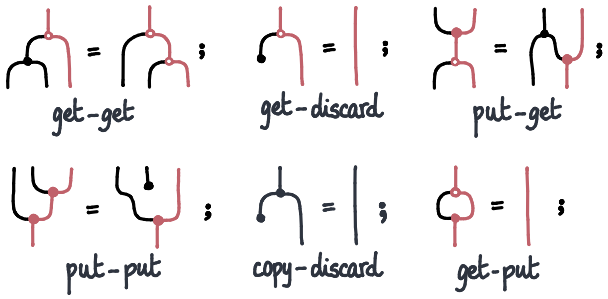


FIGURE 39. Axioms of the theory of global state.

The equations in Figure 39 say that: *(i)* reading the global state twice gets us the same result, *(ii)* reading the global state and discarding the result is the same as doing nothing, *(iii)* writing something to the global state and then reading it is the same as keeping a copy of it, *(iv)* writing twice to the global state keeps only the last thing that was written, *(v)* copying and discarding a copy is doing nothing, and *(vi)* reading something and immediately writing it to the global state is the same as doing nothing.

Proposition 1.7.11 (Race conditions)**.** *Concurrently mixing two processes that share a global state, f and g, can produce four possible results:* (i) *only the result of the first one is preserved,* (ii) *only the result of the second one is preserved, or* (iii,iv) *the composition of both is preserved, in any order, $f \fatsemi g$ or $g \fatsemi f$.*

PROOF. We work in the theory of global state adding two processes, $f\colon X \to X$ and $g\colon X \to X$, that can moreover be discarded, meaning $f \fatsemi \varepsilon = g \fatsemi \varepsilon = \varepsilon$. We employ the string diagrams of premonoidal categories. The first three diagrams in Figure 40 correspond to the first three cases, the last one is analogous to the third one. □

1.7.4. Bibliography. Alan Jeffrey pioneered a string diagrammatic representation of programs using an extra wire to represent runtime [Jef97a], all the credit for this idea should go there. Staton and Møgelberg [MS14] already showed how any premonoidal category could be reinterpreted as the Kleisli category of a state promonad.

We take these ideas a step further, showing that a particular syntax for monoidal categories can be used to talk about premonoidal categories as well. Our study may demystify premonoidal categories, or at least make them more accessible to a category theorist already interested in monoidal categories: premonoidal categories are simply monoidal categories with a hidden state object.

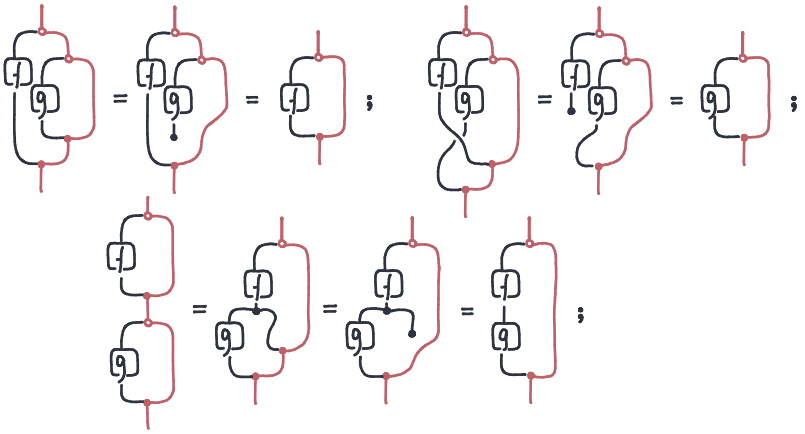


FIGURE 40. Race conditions in the theory of global state.

CHAPTER 2

Context Theory

Context Theory

Our goal in this thesis is to study how processes compose from their constituent parts while keeping for each one of these incomplete parts (each context) a meaning of its own: the meaning – the semantics – of the whole process is then determined by the semantics of each one of its parts, and how they compose. This is the principle of compositionality.

In the categorical framework, the structures that govern composition and decomposition are Lambek's multicategories: categories where morphisms have multiple inputs that get composed into a single output. We will decompose multicategories and compare them to monoidal categories using profunctors and dinaturality, which we claim to be the right mathematical tools to talk about abstract process composition in Section 2.1.

We give a short exposition of multicategories in Section 2.2, and then introduce a type of multicategory with the particular property that every transformation can be decomposed into smaller ones: *malleable multicategories*, which we prove equivalent to promonoidal categories in Section 2.3. Our main result in this section is a characterization of the multicategory that governs how morphisms compose in a category – the malleable multicategory of *spliced arrows* – as cofreely generated by the category, Section 2.4. This result is a variant on a recent result by Melliés and Zeilberger [MZ22]; and Chapter 3 will extend this theory for the first time to the setting of monoidal categories.

2.1. Profunctors and Coends

2.1.1. Profunctors. A profunctor from a category $\mathbb{A}$ to a category $\mathbb{B}$ is a functor $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$ [Bén00]. Profunctors describe families of processes indexed functorially by the objects of two different categories. The canonical example of a profunctor is the one that returns the set of morphisms between two objects of the same category, $\mathbb{A}(\bullet;\bullet)\colon \mathbb{A}^{op} \times \mathbb{A} \to \mathbf{Set}$. Profunctors, however, do not need to be restricted to a single category: this makes them useful to study the relation between processes of different categories.

Categorically, profunctors can be seen as a categorification of the concept of *relations*, functions $A \times B \to 2$. Under this analogy, existential quantifiers correspond to *coends*. This section will first introduce profunctors (Definition 2.1.1), then a naturality relation for them (Definition 2.1.4) and, finally, their composition using coends (Sections 2.1.3 and 2.1.4). We connect them explicitly to process theories in Section 2.1.5.

Definition 2.1.1. A *profunctor* $(P, \prec, \succ)$ between two categories, $\mathbb{A}$ and $\mathbb{B}$, is a family of sets, $P(A, B)$, indexed by objects $A \in \mathbb{A}_{obj}$ and $B \in \mathbb{B}_{obj}$, and endowed with jointly functorial left and right actions of the morphisms of the two categories $\mathbb{A}$ and $\mathbb{B}$, respectively.

Explicitly, types of these actions are

$$(\succ)\colon \mathbb{A}(A';A) \times P(A';B) \to P(A;B)$$
$$(\prec)\colon P(A;B) \times \mathbb{B}(B;B') \to P(A;B')$$

These two actions must be compatible, $(f \succ p) \prec g = f \succ (p \prec g)$, they must preserve identities, $id \succ p = p$, and $p \prec id = p$, and they must preserve composition $(p \prec f) \prec g = p \prec (f \fatsemi g)$ and $f \succ (g \succ p) = (f \fatsemi g) \succ p$.

More succinctly, a profunctor $P\colon \mathbb{A} \to \mathbb{B}$ is the same as a functor $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$. When presented as a family of sets with a pair of actions, profunctors have been sometimes called *bimodules*.

Definition 2.1.2 (Parallel composition)**.** Two profunctors $P\colon \mathbb{A}_1^{op} \times \mathbb{B}_1 \to \mathbf{Set}$ and $Q\colon \mathbb{A}_2^{op} \times \mathbb{B}_2 \to \mathbf{Set}$ compose *in parallel* into a profunctor $P \times Q\colon \mathbb{A}_1^{op} \times \mathbb{A}_2^{op} \times \mathbb{B}_1 \times \mathbb{B}_2 \to \mathbf{Set}$ defined by

$$(P \times Q)(A, A'; B, B') = P(A;B) \times Q(A';B').$$

Remark 2.1.3. We will consider profunctors between product categories explicitly: a *profunctor* $P\colon \mathbb{A}_0 \times ... \times \mathbb{A}_n \to \mathbb{B}_0 \times ... \times \mathbb{B}_m$ is a functor

$$P\colon \mathbb{A}_0^{op} ... \times \mathbb{A}_n^{op} \times \mathbb{B}_0 \times ... \times \mathbb{B}_m \to \mathbf{Set}.$$

For our purposes, a profunctor $P(A_0, ..., A_n; B_0, ..., B_m)$ is a family of processes indexed by contravariant inputs $A_0, ..., A_n$ and covariant outputs $B_0, ..., B_m$. The profunctor is endowed with jointly functorial left $(\succ_0, ..., \succ_n)$ and right $(\prec_0, ..., \prec_m)$ actions of the morphisms of $\mathbb{A}_0, ..., \mathbb{A}_n$ and $\mathbb{B}_0, ..., \mathbb{B}_m$, respectively [Bén00, Lor21]. We will simply use $(\prec/\succ)$ without any subscript whenever the input/output is unique.

Composing profunctors *sequentially* is subtle: the same processes could arise as the composite of different pairs of processes, so we need to impose an equivalence relation. Imagine we try to connect two different processes:

$$p \in P(A_0, ..., A_n; B_0, \dots, B_m), \text{ and } q \in Q(C_0, ..., C_k; D_0, \dots, D_h);$$

and we have some morphism $f\colon B_i \to C_j$ that translates the i-th output port of p to the j-th input port of q. Let us write $({}_i|_j)$ for this connection operation. Note that we could connect them in two different ways: we could

(1) change *the output of the first process* $p \prec_i f$ before connecting both, thus obtaining $(p \prec_i f) \,{}_i|_j\, q$;
(2) or change *the input of the second process* $f \succ_j q$ before connecting both, thus obtaining $p \,{}_i|_j\, (f \succ_j q)$.

These are different descriptions, made up of two different components. However, they essentially describe the same process: they are *dinaturally equal*. Indeed, profunctors are canonically endowed with this notion of equivalence [Bén00, Lor21], precisely equating these two descriptions. Profunctors, and their elements, are thus composed *up to dinatural equivalence*.

2.1.2. Dinaturality and Composition. Dinaturality is a canonical notion of equivalence for profunctors: it arises naturally from the construction of the bicategory of profunctors, but it also has a good interpretation in terms of procesess.

Definition 2.1.4 (Dinatural equivalence)**.** For any functor $P\colon \mathbb{C}^{op} \times \mathbb{C} \to \mathbf{Set}$, consider the set

$$S_P = \sum_{M \in \mathbb{C}} P(M; M).$$

Dinatural equivalence, $(\sim)$, on the set S_P is the smallest equivalence relation satisfying $(r \succ p) \sim (p \prec r)$ for each $p \in P(M; N)$ and each $r \in \mathbb{C}(N; M)$.

Coproducts quotiented by dinatural equivalence construct a particular form of colimit called a *coend*. Under the process interpretation of profunctors, taking a coend means *plugging an output to an input* of the same type.

Definition 2.1.5 (Coend)**.** Let $P\colon \mathbb{C}^{op} \times \mathbb{C} \to \mathbf{Set}$ be a functor. Its *coend* is the coproduct of $P(M, M)$ indexed by $M \in \mathbb{C}$, quotiented by dinatural equivalence.

$$\int^{M \in \mathbb{C}} P(M; M) \coloneqq \left(\sum_{M \in \mathbb{C}} P(M; M) \Big/ \sim \right).$$

That is, the coend is the colimit of the diagram containing a *cospan* $P(M; M) \leftarrow P(M; N) \to P(N; N)$ for each $f\colon N \to M$.

Definition 2.1.6 (Sequential composition)**.** Two profunctors $P\colon \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$ and $Q\colon \mathbb{B}^{op} \times \mathbb{C} \to \mathbf{Set}$ compose sequentially into a profunctor $P \diamond Q\colon \mathbb{A}^{op} \times \mathbb{C} \to \mathbf{Set}$ defined by

$$(P \diamond Q)(A; C) = \int^{B \in \mathbb{B}} P(A; B) \times Q(B; C).$$

The hom-profunctor $\hom\colon \mathbb{A}^{op} \times \mathbb{A} \to \mathbf{Set}$ that returns the set of morphisms between two objects is the unit for sequential composition. Sequential composition is associative up to isomorphism.

2.1.3. Coend Calculus. *Coend calculus* is the name given to the algebraic manipulations of coends that prove isomorphisms or construct natural transformations between profunctors using the behaviour of *coends*. MacLane [ML71] and Loregian [Lor21] give presentations of coend calculus.

Proposition 2.1.7 (Yoneda reduction)**.** *Let* $\mathbb{C}$ *be any category and let* $F\colon \mathbb{C} \to \mathbf{Set}$ *be a functor; the following isomorphism holds for any given object* $A \in \mathbb{C}_{obj}$*.*

$$\int^{X \in \mathbb{C}} \mathbb{C}(X; A) \times FX \cong FA.$$

Following the analogy with classical analysis, the hom *profunctor works as a Dirac's delta.*

Proposition 2.1.8 (Fubini rule)**.** *Coends commute between them; that is, there exists a natural isomorphism*

$$\int^{X_1\in\mathbb{C}}\int^{X_2\in\mathbb{C}} P(X_1,X_2;X_1,X_2) \cong \int^{X_2\in\mathbb{C}}\int^{X_1\in\mathbb{C}} P(X_1,X_2;X_1,X_2).$$

In fact, they are both isomorphic to the coend over the product category,

$$\int^{(X_1,X_2)\in\mathbb{C}\times\mathbb{C}} P(X_1,X_2;X_1,X_2).$$

Following the analogy with classical analysis, coends follow the Fubini rule for integrals.

2.1.4. The Point of Coend Calculus. In the same way that regular logic links relations, a coend calculus expression is a list of profunctors linked by some objects that are bound to a coend. Usually, the isomorphisms that we construct are never made explicit, and it is difficult for the reader to compute the precise map we constructed.

Fortunately, this has a straightforward solution. We propose to *point* the coends: to write an profunctorial expression, P, together with the *generic element* it computes, P_p. An expression of pointed coend calculus is a coend bounding some objects and a series of *pointed profunctors*. For instance, we may write

$$\textstyle\int^{M,N} P(A;M,N)_f \times Q(M;B)_g \times \mathbb{C}(N;C)_h, \quad \text{instead of just}$$
$$\textstyle\int^{M,N} P(A;M,N) \times Q(M;B) \times \mathbb{C}(N;C).$$

Coends quotient expressions by dinaturality, meaning that any left action on the covariant occurrence of a bounded variable can be equivalently written as a right action on its contravariant occurrence. In terms of pointed profunctors, this means that

$$\int^{N} P(A;N)_{(f\prec h)} \times Q(N;B)_g = \int^{M} P(A;M)_f \times Q(M;B)_{(h\succ g)}.$$

Proposition 2.1.9. *Let $\mathbb{C}$ be a category and let $F\colon \mathbb{C}^{op}\to \mathbf{Set}$ and $G\colon \mathbb{C}\to\mathbf{Set}$ be a presheaf and a copresheaf, respectively. The following are natural isomorphisms of pointed profunctors,*

$$\int^{X} \mathbb{C}(X;A)_f \times F(X)_h \;\cong\; F(A)_{(f\succ h)};$$
$$\int^{X} \mathbb{C}(A;X)_f \times G(X)_h \;\cong\; G(A)_{(h\prec f)}.$$

We call these isomorphisms the "pointed" Yoneda reductions.

Remark 2.1.10. Using pointed coends, any derivation does also include the computation of the isomorphism it induces. As an example, compare the following with the usual coend derivation of a cartesian lens [CEG+20],

Proposition 2.1.11. *In a cartesian monoidal category, the pairs of morphisms $\mathbb{C}(A;M\times X)$ and $\mathbb{C}(M\times Y;B)$, quotiented by dinaturality, are in bijective correspondence with the pairs of morphisms $\mathbb{C}(A;M)$ and $\mathbb{C}(M\times Y;B)$.*

Proof. A function is explicitly constructed by the following derivation.

$$\int^{M} \mathbb{C}(A;M\times X)_f \times \mathbb{C}(M\times Y;B)_g$$
$$\cong \quad (\textit{ by the adjunction } \Delta\dashv\times)$$

$$\begin{aligned}
&\int^{M} \mathbb{C}(A;M)_{(f\fatsemi\pi_1)} \times \mathbb{C}(A;X)_{(f\fatsemi\pi_2)} \times \mathbb{C}(M\times Y;B)_g \\
&\quad\cong \quad (\textit{ by pointed Yoneda lemma }) \\
&\mathbb{C}(A;X)_{(f\fatsemi\pi_2)} \times \mathbb{C}(X\times Y;B)_{((f\fatsemi\pi_1)\otimes id)\fatsemi g}.
\end{aligned}$$

The function mapping an equivalence class $[f,g]$ to $(f\fatsemi\pi_2;(f\fatsemi\pi_1)\otimes id)$ is a bijection because it has been constructed from composing bijections.

Indeed, in the first step, we have used that the adjunction $(\Delta \dashv \times)$ is given by postcomposition with projections and; in the second step, we use that the action on the last profunctor is defined as $h \succ g = (h\otimes id)\fatsemi g$. The bijection has been explicitly constructed as sending the pair $(f;g)$ to $(f\fatsemi\pi_2;((f\fatsemi\pi_1)\otimes id)\fatsemi g)$. □

2.1.5. Promonads. Promonads are to profunctors what monads are to functors: to quip, a promonad is just a monoid in the category of endoprofunctors. It may be then surprising to see that so little attention has been devoted to them, relative to their functorial counterparts. The main source of examples and focus of attention has been the semantics of programming languages [Hug00, Pat01, JHH09]. Strong monads are commonly used to give categorical semantics of effectful programs [Mog91], and the so-called *arrows* (or *strong promonads*) strictly generalize them: they coincide with our previous definition of effectful category [HJ06].

Part of the reason behind the relative unimportance given to promonads elsewhere may stem from precisely from that fact: promonads over a category can be shown in an elementary way to be equivalent to identity-on-objects functors from that category [Lor21]. The explicit proof is, however, difficult to find in the literature, and so we include it here (Theorem 2.1.14).

Under this interpretation, promonads are new morphisms for an old category. We can reinterpret the old morphisms into the new ones in a functorial way. The paradigmatic example is again that of Kleisli or cokleisli categories of strong monads and comonads. This structure is richer than it may sound, and we will explore it further during the rest of this text.

Definition 2.1.12. A *promonad* $(P,\star,{}^\circ)$ over a category $\mathbb{C}$ is a profunctor $P\colon \mathbb{C}^{op}\times\mathbb{C}\to\mathbf{Set}$ together with natural transformations for inclusion $({}^\circ)_{X,Y}\colon \mathbb{C}(X;Y)\to P(X;Y)$ and multiplication $(\star)_{X,Y}\colon P(X;Y)\times P(Y;Z)\to P(X;Z)$, and such that

i. the right action is premultiplication, $f^\circ\star p = f\succ p$;
ii. the left action is postmultiplication, $p\star f^\circ = p\prec f$;
iii. multiplication is dinatural, $p\star(f\succ q) = (p\prec f)\star q$;
iv. and multiplication is associative, $(p_1\star p_2)\star p_3 = p_1\star(p_2\star p_3)$.

Equivalently, promonads are promonoids in the double category of categories, where the dinatural multiplication represents a transformation from the composition of the profunctor P with itself.

Lemma 2.1.13 (Kleisli category of a promonad)**.** *Every promonad $(P,\star,{}^\circ)$ induces a category with the same objects as its base category, but with hom-sets given by $P(\bullet,\bullet)$, composition given by $(\star)$ and identities given by (id°). This is called its* Kleisli category, $\mathsf{Kleisli}(P)$. *Moreover, there exists an identity-on-objects functor $\mathbb{C}\to\mathsf{Kleisli}(P)$, defined on morphisms by the unit of the promonad.*

The converse is also true: every category $\mathbb{C}$ with an identity-on-objects functor from some base category $\mathbb{V}$ arises as the Kleisli category of a promonad.

Theorem 2.1.14. *Promonads over a category $\mathbb{C}$ correspond to identity-on-objects functors from the category $\mathbb{C}$. Given any identity-on-objects functor $i\colon\mathbb{C}\to\mathbb{D}$ there exists a unique promonad over $\mathbb{C}$ having $\mathbb{D}$ as its Kleisli category: the promonad given by the profunctor $\hom_{\mathbb{D}}(i(\bullet),i(\bullet))$.*

2.1.6. Bibliography. Coends, the Yoneda lemma, and their calculus, were introduced in MacLane's monograph [ML71]. A more modern presentation of coend calculus and its applications is in the work of Loregian [Lor21]. This author has also written on the importance of pointed profunctors for open diagrams [Rom20b] and collages [BR23].

2.2. Multicategories

2.2.1. Multicategories. Multicategories will provide an algebra for composing multiple pieces into one. A multicategory is like a category where every morphism has a list of inputs instead of a single one. A multicategory, $\mathbb{M}$, contains a set of objects, $\mathbb{M}_{obj}$, as a category does; but instead of a set of morphisms, $\mathbb{M}(X;Y)$, for every pair of objects $X, Y \in \mathbb{M}_{obj}$, it will have a set of *multimorphisms*,

$$\mathbb{M}(X_1, \dots, X_n; Y), \text{ for each list of objects } X_1, \dots, X_n, Y \in \mathbb{M}_{obj}.$$

As in sequent logic, it is easier to denote lists of objects by metavariables. For instance, we will use $\Gamma = X_1, \dots, X_n$ and write $\mathbb{M}(\Gamma; Y)$ for the set of multimorphisms $\mathbb{M}(X_1, \dots, X_n; Y)$.

Definition 2.2.1. A *multicategory*, $\mathbb{M}$, is a collection of objects, $\mathbb{M}_{obj}$, together with a collection of multimorphisms, $\mathbb{M}(\Gamma; Y)$, for each list of objects $\Gamma = X_0, \dots, X_n \in \mathbb{M}_{obj}$ and each object $Y \in \mathbb{M}_{obj}$.

For each object X, there must be an identity multimorphism, $\mathrm{id}_X \in \mathbb{M}(X; X)$. For each three lists of objects $\Gamma, \Gamma_1, \Gamma_2$ and each two objects Y and Z, there must exist a composition operation (we omit superscripts when clear from the context),

$$(\fatsemi)_i^{\Gamma_1,\Gamma_2} \colon \mathbb{M}(\Gamma; Y_i) \times \mathbb{M}(\Gamma_1, Y_i, \Gamma_2; Z) \to \mathbb{M}(\Gamma_1, \Gamma, \Gamma_2; Z).$$

Composition must be unital, meaning that $\mathrm{id}_X \fatsemi_X f = f$ and $f \fatsemi \mathrm{id}_Y = f$ every time that the equation is fomally well-typed. Composition must be also associative, meaning that $(h \fatsemi_X g) \fatsemi_Y f = h \fatsemi_X (g \fatsemi_Y f)$; and $g \fatsemi_Y (h \fatsemi_X f) = h \fatsemi_X (g \fatsemi_Y f)$ must hold whenever they are formally well-typed, see Figure 1.

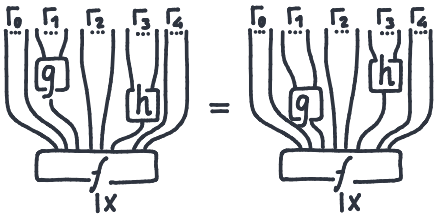


FIGURE 1. Associativity for a multicategory.

2.2.2. The Category of Multicategories. In the same way categories are the first step towards the theory of functors and natural transformations, multicategories are the first step towards the theory of multifunctors and multinatural transformations. In the same way the formal theory of categories is synthetised by the 2-category **Cat** of categories, functors and natural transformations; the study of multicategories is synthetised by the 2-category **MultiCat** of multicategories, multifunctors and multinatural transformations.

Definition 2.2.2. A *multifunctor* between two multicategories, $F \colon \mathbb{M} \to \mathbb{N}$, consists of an assignment on objects $F_{obj} \colon \mathbb{M} \to \mathbb{N}$ and an assignment on multimorphisms of any arity,

$$F_n \colon \mathbb{M}(X_1, \dots, X_n; Y) \to \mathbb{N}(F_{obj}X_1, \dots, F_{obj}X_n; F_{obj}Y),$$

that preserves identities, $F_1(\mathrm{id}_X) = \mathrm{id}_{F_{obj}(X)}$, and composition of multimorphisms, $F_{n+m-1}(f \fatsemi_Y g) = F_n(f) \fatsemi_{F_{obj}(Y)} F_m(g)$.

Definition 2.2.3. A multinatural transformation $\theta\colon F \to G$ between two multifunctors $F, G\colon \mathbb{M} \to \mathbb{N}$ is given by a family of multimorphisms $\theta_X \in \mathbb{N}(FX; GX)$ such that, for each multimorphism $f \in \mathbb{M}(X_1, \dots, X_n; Y)$, the following naturality condition holds

$$\theta_{X_1} \fatsemi_1 \dots \fatsemi_{n-1} \theta_{X_n} \fatsemi G(f) = F(f) \fatsemi \theta_Y.$$

Proposition 2.2.4. *Multicategories with multifunctors between them form a category,* **Mult**.

2.2.3. Application: Shufflings. Let us exemplify multicategories with an example that will become increasingly relevant in this thesis. Shufflings are permutations that preserve the relative ordering of some blocks. We can always count shufflings combinatorially, but multicategories provide the extra structure that allows us to track how different shufflings compose.

Example 2.2.5. A *shuffling* is a permutation of the elements of multiple blocks that preserves their relative ordering. The multicategory of shufflings has objects the natural numbers and morphisms the shufflings, $\sigma \in \mathbf{Shuf}(p_0, \dots, p_n; q)$ that reorganize $p_0, \dots, p_n$ elements into $q = p_0 + \dots + p_n$ without altering their internal ordering.

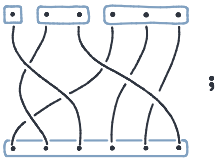


FIGURE 2. Example of a 1,2,3-shuffling.

More explicitly, the number of shufflings $\mathbf{Shuf}(p_0, \dots, p_n; q)$ is given by a multinomial coefficient whenever $q = p_0 + \dots + p_n$,

$$\#\mathbf{Shuf}(p_0, \dots, p_n; p_0 + \dots + p_n) = \frac{(p_0 + \dots + p_n)!}{p_0! \cdot \dots \cdot p_n!},$$

and it is zero in any other case.

Shufflings exhibit a particular property that motivates our next section: *malleability*. Any shuffling of p_0, p_1 and p_2 can be factored *uniquely* in two different forms: we can first shuffle p_0 and p_1 and then shuffle the result, $p_0 + p_1$, with p_2; or we can first shuffle p_1 and p_2, and then shuffle p_0 with the result, $p_1 + p_2$. For instance, any $1, 2, 3$-shuffling splits uniquely into a $2, 3$-shuffling followed by a $1, 5$-shuffling, but also uniquely into a $1, 2$-shuffling followed by a $3, 3$-shuffling.

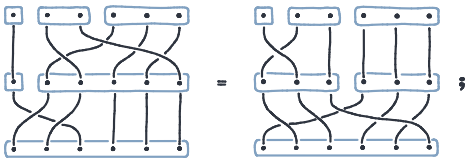


FIGURE 3. Two factorizations of the previous shuffling.

This is a global property: any shuffling can be uniquely factored into smallest shufflings, in whichever arrangement we pick. Any morphism of the multicategory **Shuf** can be factored into any possible shape, uniquely. We say that the multicategories that satisfy this property are *"malleable multicategories"*: shufflings form a malleable multicategory.

2.3. Malleable Multicategories

A malleable multicategory is a multicategory where each morphism can be morphed uniquely into any possible shape. This means that there exist unique factorizations of each morphism into each one of the possible shapes. Formally, we will define malleable multicategories to have an invertible composition, up to the morphisms of some underlying category.

Definition 2.3.1. The unary morphisms of a multicategory form a category [Shu16]. In other words, given a multicategory $\mathbb{M}$, the underlying category, $\mathbb{M}^u$, has the same objects as the multicategory, $\mathbb{M}^u_{obj} = \mathbb{M}_{obj}$, and morphisms defined from the unary multimorphisms of the multicategory, $\mathbb{M}^u(X;Y) = \mathbb{M}(X;Y)$. Composition and identities are exactly those of the multicategory.

Remark 2.3.2. The multimorphisms of a multicategory determine profunctors over the underlying category of the multicategory. The underlying category acts on the multimorphisms by composition,

$$(\succ)\colon \mathbb{M}^u(X;X') \times \mathbb{M}(\Gamma_1, X', \Gamma_2; Y) \to \mathbb{M}(\Gamma_1, X, \Gamma_2; Y),$$
$$(\prec)\colon \mathbb{M}(\Gamma; Y) \times \mathbb{M}^u(Y;Y') \to \mathbb{M}(\Gamma; Y').$$

In any multicategory, composition of multimorphisms is dinatural with respect to the underlying category. This follows from the associativity for multicategories,

$$(f \prec h) \fatsemi_{X_i} g = (f \fatsemi_{X_i} h) \fatsemi_{X_i'} g = f \fatsemi_{X_i} (h \fatsemi_{X_i'} g) = f \fatsemi_{X_i} (h \succ g).$$

As a consequence, composition is well-defined under dinaturality. We define dinatural composition to be composition lifted to the equivalence classes of the dinaturality equivalence relation, which are written as a coend,

$$(\fatsemi)\colon \left(\int^{Y \in \mathbb{M}} \mathbb{M}(\Gamma; Y) \times \mathbb{M}(\Gamma_0, Y, \Gamma_1; Z)\right) \to \mathbb{M}(\Gamma_0, \Gamma, \Gamma_1; Z).$$

Definition 2.3.3. A *malleable multicategory* is a multicategory where dinatural composition is invertible.

Proposition 2.3.4. *Malleable multicategories with multifunctors between them form a category,* **mMult**. *This is a wide subcategory of the category of multicategories.*

Remark 2.3.5. If a multicategory is malleable, we can reconstruct it up to isomorphism from its binary and nullary maps. When defining a malleable multicategory, it is usually easier to provide its binary, unary and nullary maps, and deduce from those the rest of the structure. The situation is similar in monoidal categories: we do not need to provide the n-ary tensor in order to define a monoidal category, we only provide the binary and unary tensors.

This suggests that we will really work with a *biased* version of malleable multicategories, one that privileges the binary and nullary tensors over the others. Biased malleable multicategories are better known as *promonoidal categories*.

2.3.1. Promonoidal Categories. In the same sense that multicategories provide an algebra for the composition of multiple pieces into one, promonoidal categories provide an algebra for the *coherent composition* of multiple pieces into one. A category $\mathbb{C}$ contains sets of *morphisms*, $\mathbb{C}(X;Y)$. In the same way, a promonoidal category $\mathbb{V}$ contains sets of *joints*, $\mathbb{V}(X_0 \lhd X_1; Y)$, *morphisms*, $\mathbb{V}(X;Y)$, and *units*, $\mathbb{V}(N;X)$, where N is the virtual tensor unit. Joints, $\mathbb{V}(X_0 \lhd X_1; Y)$, represent a way of joining objects of type X_0 and X_1 into an objects of type Y. Morphisms, $\mathbb{V}(X;Y)$, as in any category, are transformations of X into Y. Units, $\mathbb{V}(N;Y)$, are the atomic pieces of type Y.

These compositions must now be coherent. For instance, imagine we want to join X_0, X_1 and X_2 into Y. Joining X_0 and X_1 into something ($\bullet$), and then joining that something ($\bullet$) and X_2 into Y, *should be doable in essentially the same ways* as joining X_1 and X_2 into something ($\bullet$), and then joining X_0 and that something ($\bullet$) into Y. Formally, we are saying that,

$$\int^U \mathbb{V}(X_0 \triangleleft X_1; U) \times \mathbb{V}(U \triangleleft X_2; Y) \cong \int^V \mathbb{V}(X_1 \triangleleft X_2; V) \times \mathbb{V}(X_0 \triangleleft V; Y),$$

and, in fact, we usually just write $\mathbb{V}(X_0 \triangleleft X_1 \triangleleft X_2; Y)$ for the set of such decompositions, even when it is only defined up to isomorphism.

Definition 2.3.6. Promonoidal categories are the 2-monoids of the monoidal bicategory of profunctors, which is equivalent to the following definition. A *promonoidal category* is a category $\mathbb{V}(\bullet;\bullet)$ endowed with two profunctors

$$\mathbb{V}(\bullet \triangleleft \bullet; \bullet) \colon \mathbb{V} \times \mathbb{V} \to \mathbb{V}, \text{ and } \mathbb{V}(\mathsf{N}; \bullet) \colon 1 \to \mathbb{V}.$$

Equivalently, these are functors

$$\mathbb{V}(\bullet \triangleleft \bullet; \bullet) \colon \mathbb{V}^{op} \times \mathbb{V} \times \mathbb{V} \to \mathbf{Set}, \text{ and } \mathbb{V}(\mathsf{N}; \bullet) \colon \mathbb{V}^{op} \to \mathbf{Set}.$$

Moreover, promonoidal categories must be endowed with the following natural isomorphisms,

$$\begin{aligned} \mathbb{V}(X_0 \triangleleft X_1; \bullet) \diamond \mathbb{V}(\bullet \triangleleft X_2; Y) &\cong \mathbb{V}(X_1 \triangleleft X_2; \bullet) \diamond \mathbb{V}(X_0 \triangleleft \bullet; Y); \\ \mathbb{V}(\mathsf{N}; \bullet) \diamond \mathbb{V}(\bullet \triangleleft X; Y) &\cong \mathbb{V}(X; Y); \\ \mathbb{V}(\mathsf{N}; \bullet) \diamond \mathbb{V}(X \triangleleft \bullet; Y) &\cong \mathbb{V}(X; Y); \end{aligned}$$

called α, λ, ρ, respectively, and asked to satisfy the pentagon and triangle coherence equations, $\alpha \fatsemi \alpha = (\alpha \diamond \mathrm{id}) \fatsemi \alpha \fatsemi (\mathrm{id} \diamond \alpha)$, and $(\rho \diamond \mathrm{id}) = \alpha \fatsemi (\lambda \diamond \mathrm{id})$.

Definition 2.3.7 (Promonoidal functor)**.** Let $\mathbb{V}$ and $\mathbb{W}$ be two promonoidal categories. A *promonoidal functor* $F \colon \mathbb{V} \to \mathbb{W}$ is a functor between the two categories together with natural transformations

$$F_\triangleleft \colon \mathbb{V}(X_0 \triangleleft X_1; Y) \to \mathbb{W}(FX_0 \triangleleft FX_1; FY), \quad \text{and} \quad F_\mathsf{N} \colon \mathbb{V}(\mathsf{N}; X) \to \mathbb{W}(\mathsf{N}; Y),$$

that satisfy $\lambda \fatsemi F_{map} = (F_\triangleleft \times F_N) \fatsemi \lambda$, $\rho \fatsemi F_{map} = (F_\triangleleft \times F_N) \fatsemi \rho$, and $\alpha \fatsemi (F_\triangleleft \times F_\triangleleft) \fatsemi i = (F_\triangleleft \times F_\triangleleft) \fatsemi i \fatsemi \alpha$.

Proposition 2.3.8. *Promonoidal categories with promonoidal functors between them form a category,* **Prom**.

2.3.2. Promonoidal Categories are Malleable Multicategories. In this section, we show that the category of promonoidal categories is equivalent to that of malleable multicategories. In this sense, the study of malleable multicategories is the study of promonoidal categories.

Definition 2.3.9 (Underlying malleable multicategory)**.** Let $\mathbb{V}$ be a promonoidal category. There is a malleable multicategory, $\mathbb{V}^m$, that has the same objects but multimorphisms defined by the elements of the promonoidal category. By induction, we define

$$\begin{aligned} \mathbb{V}^m(X_0, X_1, \Gamma; Y) &= \textstyle\int^V \mathbb{V}(X_0 \triangleleft X_1; V) \times \mathbb{V}^m(V, \Gamma; Y), \\ \mathbb{V}^m(X; Y) &= \mathbb{V}(X; Y), \\ \mathbb{V}^m(; Y) &= \mathbb{V}(\mathsf{N}; Y). \end{aligned}$$

In other words, the multimorphisms are elements of the left-biased tree reductions of the promonoidal category, seen as a 2-monoid. Dinatural composition is then

defined to be the unique map relating two tree expressions in a 2-monoid, which exists uniquely by coherence,

$$(\text{coh})\colon \left(\int^{X\in\mathbb{V}} \mathbb{V}(\Gamma; X) \times \mathbb{V}^m(\Gamma_0, X, \Gamma_1; Y)\right) \to \mathbb{V}^m(\Gamma_0, \Gamma, \Gamma_1; X).$$

Coherence maps are isomorphisms, and so dinatural composition is invertible, making the multicategory coherent. By coherence for pseudomonoids, composition must satisfy associativity and unitality.

Proposition 2.3.10. *The category of promonoidal categories and the category of malleable multicategories are equivalent with the functor* $(\bullet)^m\colon \mathbf{Prom} \to \mathbf{mMult}$ *induced by the construction of the underlying malleable multicategory of a promonoidal category. See a polycategorical analogue at Proposition 0.2.16.*

Proof. First, let us show that a promonoidal functor, $F\colon \mathbb{V} \to \mathbb{W}$, induces a multifunctor, $F^m\colon \mathbb{V}^m \to \mathbb{W}^m$ between the underlying multicategories. On objects, we define it to be the same, $F^m_{obj} = F_{obj}$. On multimorphisms, we define the binary, unary and nullary cases using the promonoidal transformations:

$$F^u_0 = F_N, \text{ with } F^u_1 = F_{map} \text{ and } F^u_2 = F_\triangleleft.$$

Then, we extend this definition to the n-ary case by induction, using $F^u_n = F_\triangleleft \times F^u_{n-1}$. We now verify that this assignment is functorial: the only remarkable case is that of checking that the inductive case preserves composition.

$$\begin{aligned}(F \fatsemi G)^u_n &\overset{(i)}{=} (F \fatsemi G)_\triangleleft \times (F \fatsemi G)^u_{n-1} \overset{(ii)}{=} (F_\triangleleft \fatsemi G_\triangleleft) \times (F^u_{n-1} \fatsemi G^u_{n-1}) \\ &\overset{(iii)}{=} (F_\triangleleft \times F^u_{n-1}) \times (G_\triangleleft \times G^u_{n-1}) \overset{(iv)}{=} F^u_n \fatsemi G^u_n.\end{aligned}$$

This equation holds because *(i)* of the inductive definition, *(ii)* the composition of promonoidal functors and the inductive hypothesis, *(iii)* the interchange law for functions, *(iv)* and the inductive definition.

We will now show that this is a fully faithful functor. It is hopefully clear that it is faithful because $F^u = G^u$ directly implies $F_{obj} = G_{obj}$, $F_\triangleleft = G_\triangleleft$, $F_N = G_N$ and $F_{map} = G_{map}$. Let us show that it is also full. Let $G\colon \mathbb{V}_u \to \mathbb{W}_u$ be a multifunctor between malleable multicategories. We will construct a promonoidal functor $G^\star$ such that $G^{\star u} = G$. We start by defining that $G^\star_N = G_0$, that $G^\star_{map} = G_1$ and that $G^\star_\triangleleft = G_2$. Now, we need to prove that $(G^{\star u})_n = G_n$. This is definitionally true for binary, unary and nullary multimorphisms; then, by induction,

$$G_n \overset{(i)}{=} \text{decomp} \fatsemi (G_2 \times G_n) \overset{(ii)}{=} \text{decomp} \fatsemi (G^{\star u}_2 \times G^{\star u}_n) \overset{(iii)}{=} G^{\star u}_n.$$

Here, we use *(i)* malleability, *(ii)* the induction hypothesis, and *(iii)* the definition of the underlying multifunctor. We have shown that we have a fully faithful functor.

Finally, we will show that $U\colon \mathbf{Prom} \to \mathbf{mMult}$ is essentially surjective. Given any malleable multicategory $\mathbb{M}$, we can define $M^\flat$ to be the promonoidal category with the same objects and only binary, unary and nullary morphisms. We now note that there is an isomorphism, $\mathbb{M} \cong \mathbb{M}^{\flat u}$, that is the identity on objects. It is defined to use the invertible dinatural composition that exists by malleability,

$$\begin{aligned}\mathbb{M}^u_\flat(X_1, \ldots, X_n; Y) &= \int^U \mathbb{M}(X_0, X_1; U) \times \mathbb{M}(U, X_2, \ldots, X_n; Y) \\ &\cong \mathbb{M}(X_1, \ldots, X_n; Y).\end{aligned}$$

This makes the functor fully faithful and essentially surjective, defining an equivalence of categories. □

2.3.3. Bibliography. What makes monoidal tensors universal? Products and coproducts have a universal property, but that is the exception and not the rule. Hermida's work [Her00] explains that tensors are universal because they represent a relevant multimap structure, a *multicategory*. For instance, the monoidal category of vector spaces with their tensor product represents functions linear in each variable: a linear function $A \otimes B \to C$ is the same as a multilinear function $A, B \to C$. Indeed, Lambek [Lam69] first introduced multicategories as the underlying structure that unified Gentzen's sequents and multilinear maps.

2.4. The Splice-Contour Adjunction

2.4.1. Contour of a multicategory. This last section characterizes the cofree malleable multicategory of *spliced arrows*, which governs how incomplete morphisms can be nested. Any multicategory freely generates another category, its *contour* [MZ22]. This can be interpreted as the category that tracks the processes of decomposition that the multicategory describes. The construction is particularly pleasant from the geometric point of view: it takes its name from the fact that it can be constructed by following the contour of the shape of the decomposition (Figure 4).

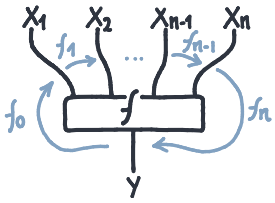


FIGURE 4. Contour of a multimorphism.

Definition 2.4.1. Let $\mathbb{M}$ be a multicategory. Its contour, $\mathsf{Contour}(\mathbb{M})$, is the category presented by two polarized objects, X° and $X^\bullet$, for each object $X \in \mathbb{M}_{obj}$;

(1) for each multimorphism $f \in \mathbb{M}(X_1, \dots, X_n; Y)$, the following generators,

$$f_0 \colon Y^\circ \to X_1^\circ; f_1 \colon X_1^\bullet \to X_2^\circ; \quad \dots; \quad f_{n-1} \colon X_{n-1}^\bullet \to X_n^\circ; f_n \colon X_n^\bullet \to Y^\bullet;$$

with only an $f_0 \colon Y^\circ \to Y^\bullet$ for the case $n = 0$;

(2) requiring contour to preserve identities, $(\mathrm{id}_X)_0 = \mathrm{id}_{X^\circ}$ and $(\mathrm{id}_X)_1 = \mathrm{id}_{X^\bullet}$;

(3) and requiring contour to preserve compositions, meaning that for each $f \in \mathbb{M}(X_1, \dots, X_n; Y_i)$ and each $g \in \mathbb{M}(Y_1, \dots, Y_m; Z)$, the contour of their composition is defined by the following five cases

$$(f \fatsemi_{X_i} g)_j = \begin{cases} g_j & \text{when } j < i, \\ g_i \fatsemi f_0 & \text{when } j = i, \\ f_{j-i} & \text{when } i < j < i + n, \\ f_n \fatsemi g_{i+1} & \text{when } j = i + n, \\ g_{j-n+1} & \text{when } i + n < j < n + m, \end{cases}$$

with the special case $(f \fatsemi_{X_i} g)_i = g_i \fatsemi f_0 \fatsemi g_{i+1}$ whenever $n = 0$.

A recent article by Melliès and Zeilberger [MZ22] develops the notion of a context-free grammar over a category as a multicategorical functor to the multicategory of *spliced arrows*. The multicategory of spliced arrows is a universal construction over a category that produces a multicategory of "contexts" over the category.

2.4.2. Spliced Arrows. The multicatgeory of spliced arrows is formed by arrows containing *blanks* or *holes* that could be filled to constuct a morphism. This multicategory gives an algebraic theory of context for the category.

Definition 2.4.2 (Spliced arrows)**.** Let $\mathbb{C}$ be a category. The multicategory of *spliced arrows* has objects pairs of objects in $\mathbb{C}$, and the multimorphisms are given by sequences of arrows in $\mathbb{C}$ separated by n gaps

$$\mathsf{Splice}\mathbb{C}\left(\begin{smallmatrix} X_1 \\ Y_1 \end{smallmatrix}, \dots, \begin{smallmatrix} X_n \\ Y_n \end{smallmatrix}; \begin{smallmatrix} X \\ Y \end{smallmatrix}\right) = \mathbb{C}(X; X_1) \times \prod_{k=1}^{n-1} \mathbb{C}(Y_k, X_{k+1}) \times \mathbb{C}(Y_n; Y);$$

which we write as $f_0 \fatsemi \Box \fatsemi \ldots \fatsemi \Box \fatsemi f_n$. That is, the sequence goes from X to Y, with holes typed by $\{X_i \to Y_i\}_{0<i\leq n}$. Composition is defined by substitution

$$(f_0 \fatsemi \Box \fatsemi \ldots \fatsemi \Box \fatsemi f_n) \succ_i (g_0 \fatsemi \Box \fatsemi \ldots \fatsemi \Box \fatsemi g_m) = \\ (g_0 \fatsemi \Box \fatsemi \ldots \fatsemi g_i \fatsemi f_0 \fatsemi \Box \fatsemi \ldots \fatsemi \Box \fatsemi f_n \fatsemi g_{i+1} \fatsemi \ldots \fatsemi \Box \fatsemi g_m),$$

and the identity morphism in $\binom{X}{Y}$ is $id_X \fatsemi \Box \fatsemi id_Y$.

Proposition 2.4.3. *The multicategory of spliced arrows is a malleable multicategory.*

Proof. We will show that dinatural composition is invertible by exhibiting an inverse to the composition operation, up to dinaturality. Consider a spliced arrow $h_0 \fatsemi \Box \cdots \Box \fatsemi h_{n+m-1}$ with $(n+m+1)$ holes; we can decompose n of its holes at position i with the following operation.

$$\begin{aligned}&\mathrm{decomp}_i^n(h_0 \fatsemi \Box \cdots \Box \fatsemi h_{n+m-1}) \\ &\quad = (h_0 \fatsemi \Box \cdots h_i \fatsemi \Box \fatsemi h_{i+n} \cdots \Box \fatsemi h_{n+m-1}) \mid (\mathrm{id} \fatsemi \Box \fatsemi h_{i+1} \cdots h_{i+n-1} \fatsemi \Box \fatsemi \mathrm{id}) \\ &\quad = (h_0 \fatsemi \Box \cdots \mathrm{id} \fatsemi \Box \fatsemi \mathrm{id} \cdots \Box \fatsemi h_{n+m-1}) \mid (h_i \fatsemi \Box \fatsemi h_{i+1} \cdots h_{i+n-1} \fatsemi \Box \fatsemi h_{i+n}).\end{aligned}$$

This operation is an inverse to dinatural composition. It follows by construction that $(\succ)_i \fatsemi \mathrm{decomp}_i = \mathrm{id}$, and we now check that $\mathrm{decomp}_i \fatsemi (\succ)_i$ is an identity up to dinaturality. Let $f_0 \fatsemi \Box \cdots \Box \fatsemi f_n$ and $g_0 \fatsemi \Box \cdots \Box \fatsemi g_m$ be two spliced arrows,

$$\begin{aligned}&\mathrm{decomp}_i^n((f_0 \fatsemi \Box \ldots \Box \fatsemi f_n) \succ_i (g_0 \fatsemi \Box \ldots \Box \fatsemi g_m)) \\ &= \mathrm{decomp}_i^n(g_0 \fatsemi \Box \ldots \fatsemi g_i \fatsemi f_0 \fatsemi \Box \fatsemi \ldots \Box \fatsemi f_n \fatsemi g_{i+1} \fatsemi \Box \cdots \Box \fatsemi g_m)) \\ &= (g_0 \fatsemi \ldots \fatsemi g_i \fatsemi f_0 \fatsemi \Box \fatsemi f_n \fatsemi g_{i+1} \fatsemi \ldots \fatsemi g_m) \mid (\mathrm{id} \fatsemi \Box \fatsemi f_1 \fatsemi \ldots \fatsemi f_{n-1} \fatsemi \Box \fatsemi \mathrm{id}) \\ &= (g_0 \fatsemi \ldots \fatsemi g_i \fatsemi \Box \fatsemi g_{i+1} \fatsemi \ldots \fatsemi g_m) \mid (f_0 \fatsemi \Box \fatsemi f_1 \fatsemi \ldots \fatsemi f_{n-1} \fatsemi \Box \fatsemi f_n).\end{aligned}$$

We have shown that dinatural composition is invertible. □

2.4.3. Splice-Contour Adjunction. A first explicit account of splice-contour adjunction is due to Melliès and Zeilberger [MZ22]. In later joint work with Earnshaw and Hefford [EHR23], we showed that this adjunction produced not only multicategories but malleable multicategories.

Theorem 2.4.4. *Splice is right adjoint to contour.*

Proof. Let $\mathbb{M}$ be a multicategory. We will show that any multifunctor to a spliced arrow multicategory, $F\colon \mathbb{M} \to \mathsf{Splice}(\mathbb{C})$, factors through a canonical multifunctor $T\colon \mathbb{M} \to \mathsf{Splice}(\mathsf{Contour}(\mathbb{M}))$ that is followed by a unique functor $F^\sharp\colon \mathsf{Contour}(\mathbb{M}) \to \mathbb{C}$. First, we construct $T\colon \mathbb{M} \to \mathsf{Splice}(\mathsf{Contour}(\mathbb{M}))$, the multifunctor that sends any object X to the pair of polarized objects $\binom{X^\circ}{X^\bullet}$; and that sends any multimap $f \in \mathbb{M}(X_0, \ldots, X_n; Y)$ to the spliced arrow

$$f_0 \fatsemi \Box \fatsemi \ldots \fatsemi \Box \fatsemi f_n \in \mathsf{Splice}(\mathbb{C}) \begin{pmatrix} X_0^\circ \\ X_0^\bullet \end{pmatrix}, \ldots, \begin{pmatrix} X_n^\circ \\ X_n^\bullet \end{pmatrix}; \begin{pmatrix} Y^\circ \\ Y^\bullet \end{pmatrix}.$$

We check now that T is indeed a multifunctor: by construction, it sends $T(f \fatsemi_{X_i} g) = f \succ_i g$ and it sends $T(\mathrm{id}_X) = (\mathrm{id}_{X^\circ} \fatsemi \Box \fatsemi \mathrm{id}_{X^\bullet})$.

We will now show that there exists a unique functor $F^\sharp\colon \mathsf{Contour}(\mathbb{M}) \to \mathbb{C}$ factoring the multifunctor F, such that $F = T \fatsemi \mathsf{Splice}(F^\sharp)$. The contour is a category presented by some generators and equations: to define a functor from it, it suffices to define it on the generators and show that it preserves the equations of the presentation. We do so next. Consider the objects, for each $X \in \mathbb{M}_{obj}$, assume $F(X) = \binom{A}{B}$. We must have that

$$\mathsf{Splice}(F^\sharp)(T(X)) = \mathsf{Splice}(F^\sharp)\binom{X^\circ}{X^\bullet} = \begin{pmatrix} F^\sharp X^\circ \\ F^\sharp X^\bullet \end{pmatrix} = \binom{A}{B},$$

which forces $F^\sharp(X^\circ) = A$ and $F^\sharp(X^\bullet) = B$. Consider now the morphisms, for each $f \in \mathbb{M}(X_0, \dots, X_n; Y)$. We must have that

$$\mathsf{Splice}(F^\sharp)(T(f)) = \mathsf{Splice}(F^\sharp)(f_0 \fatsemi \dots \fatsemi f_n) = F^\sharp f_0 \fatsemi \square \fatsemi \dots \fatsemi \square \fatsemi F^\sharp f_n,$$

which forces $F^\sharp(f_i) = F(f)_i$. This uniquely determines the value of $F^\sharp$ in all of the morphisms of the contour. We must finally check that that $F^\sharp$ satisfies the equations. We first notice that, by functoriality of F, we have

$$F^\sharp((f \fatsemi_{X_i} g)_j) = F(f \fatsemi_{X_i} g)_j = (F(f) \fatsemi_{FX_i} F(g))_j,$$

and, using this and the previous $F^\sharp(f_i) = F(f)_i$, we simply check the five cases of contour composition,

$$F^\sharp((f \fatsemi_{X_i} g)_j) = \left\{ \begin{array}{l} F(g)_j \\ F(g)_i \fatsemi F(f)_0 \\ F(f)_{j-i} \\ F(f)_n \fatsemi F(g)_{i+1} \\ F(g_{j-n+1}) \end{array} \right\} = \left\{ \begin{array}{l} F^\sharp(g_j) \\ F^\sharp(g_i) \fatsemi F^\sharp(f_0) \\ F^\sharp(f)_{j-i} \\ F^\sharp(f)_n \fatsemi F^\sharp(g)_{i+1} \\ F^\sharp(g)_{j-n+1} \end{array} \right\}.$$

This proves the existence of $F^\sharp$, but it also proves that it is the unique possible functor such that $F = T \fatsemi \mathsf{Splice}(F^\sharp)$. □

2.4.4. Promonoidal Splice-Contour. We have commented on how any malleable multicategory induces a promonoidal category. The malleable multicategory of spliced arrows induces a promonoidal category of spliced arrows. This promonoidal category is precisely the one that arises from the adjunction between any category and its opposite category in the monoidal bicategory of profunctors.

Remark 2.4.5. The promonoidal splice could be seen as a particular case of the more general multicategorical splice. However, we will see that it is usually better behaved: technically, it is the 2-monoid arising from the 2-duality of a category with its opposite category, $\mathbb{C} \dashv \mathbb{C}^{op}$, in the monoidal bicategory of profunctors. We will not use this particular fact too much, but it will inspire its generalization. In the next chapter, we repeat a *monoidal* version of the splice-contour adjunction that, by default, uses only the malleable version.

Proposition 2.4.6 (Promonoidal spliced arrows)**.** *Let $\mathbb{C}$ be a category. The promonoidal category of* spliced arrows*, $\mathsf{Splice}\mathbb{C}$, has as objects pairs of objects of $\mathbb{C}$. It uses the following profunctors to define morphisms, splits and units.*

(1) $\mathsf{Splice}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \mathbb{C}(A; X) \times \mathbb{C}(Y, B)$;
(2) $\mathsf{Splice}(\mathbb{C})(\begin{smallmatrix} X \\ Y \end{smallmatrix} \lhd \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; X) \times \mathbb{C}(Y; X') \times \mathbb{C}(Y'; B)$;
(3) $\mathsf{Splice}(\mathbb{C})(N; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; B)$.

In other words, morphisms are *pairs of arrows* $f\colon A \to X$ and $g\colon Y \to B$. Splits are *triples of arrows* $f\colon A \to X$, $g\colon Y \to X'$ and $h\colon Y' \to B$. Units are simply *arrows* $f\colon A \to B$. We use the following notation for

(1) morphisms, $(f \fatsemi \square \fatsemi g)\colon \left(\begin{smallmatrix} X \\ Y \end{smallmatrix}\right) \to \left(\begin{smallmatrix} A \\ B \end{smallmatrix}\right)$;
(2) joins, $(f \fatsemi \square \fatsemi g \fatsemi \square \fatsemi h)\colon \left(\begin{smallmatrix} X \\ Y \end{smallmatrix}\right) \lhd \left(\begin{smallmatrix} X' \\ Y' \end{smallmatrix}\right) \to \left(\begin{smallmatrix} A \\ B \end{smallmatrix}\right)$;
(3) and units, $f\colon \mathsf{N} \to \left(\begin{smallmatrix} A \\ B \end{smallmatrix}\right)$.

Definition 2.4.7. The *contour* of a promonoidal category $\mathbb{P}$ is the category $\mathsf{Contour}(\mathbb{P})$ presented by two polarized objects, X° and $X^\bullet$, for each object $X \in \mathbb{P}_{obj}$; and generated by the arrows that arise from *contouring* the decompositions of the promonoidal category.

Specifically, the contour is the category presented by the following generators, as depicted in Figure 5:

(1) $a_0\colon Y^\circ \to X^\circ$ and $a_1\colon X^\bullet \to Y^\bullet$, for each morphism $a \in \mathbb{P}(X; Y)$;

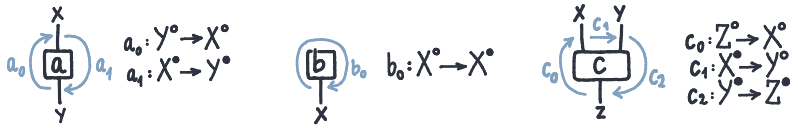


FIGURE 5. Contour of a promonoidal category.

(2) $b_0 \colon X^\circ \to X^\bullet$, for each unit $b \in \mathbb{P}(N; X)$;
(3) a triple of generators, $c_0 \colon Z^\circ \to X^\circ$, $c_1 \colon X^\bullet \to Y^\circ$ and $c_2 \colon Y^\bullet \to Z^\bullet$, for each split $c \in \mathbb{P}(X \triangleleft Y; Z)$.

We impose several equations over these generators, all depicted in Figure 6. The equations come from the decomposition of categories and they form the theory of contour.

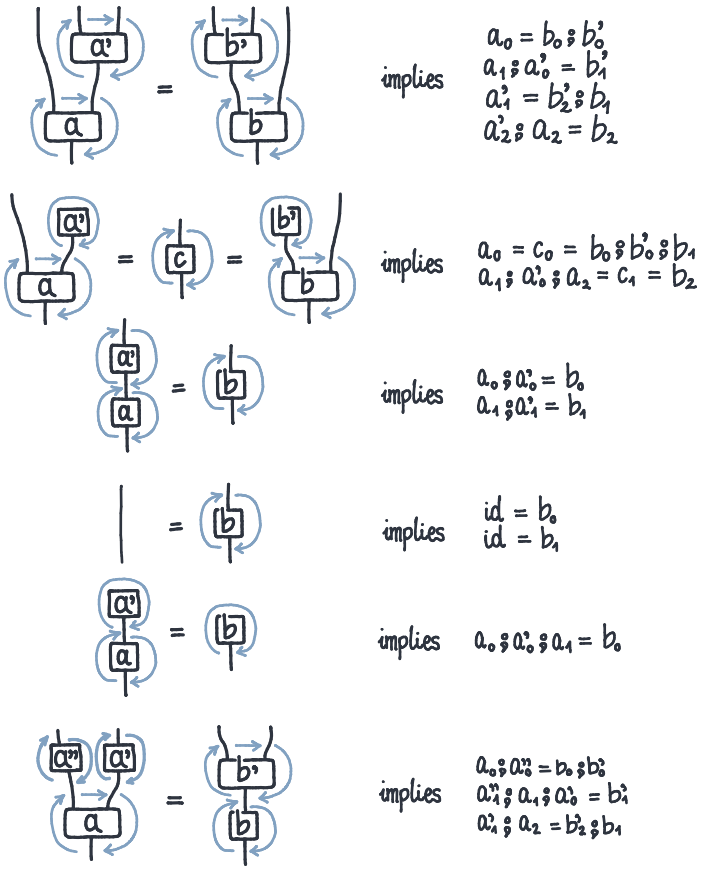


FIGURE 6. Theory of contour.

CHAPTER 3

Monoidal Context Theory

Monoidal Context Theory

This section develops a theory of context, or incomplete parts, for arbitrary monoidal categories. In the same way that the theory of context for categories required malleable multicategories or promonoidal categories; the theory of context for monoidal categories will require duoidal categories and produoidal categories.

Duoidal categories combine a sequential tensor ($\triangleleft$) with a parallel tensor ($\otimes$); and it is well known that they can be used for process description [GF16, SS22]. However, lifting context theory to monoidal categories will come with a few technical surprises that we develop; the most important one is a normalization monad in the category of produoidal categories: this becomes a crucial step in creating a theory of monoidal context that allows incomplete morphisms to take any shape. The morphisms of the produoidal category of contexts have been called *lenses* and *combs* in the literature, and we characterize them by a universal property.

We revisit the literature on duoidal categories and normalization in Sections 3.1 and 3.2. Produoidal categories and their splice-contour adjunction are introduced in Section 3.3. We take a technical aside in Section 3.4 to construct the normalization monad, and we immediately use it in Section 3.5.

3.1. Duoidal categories

3.1.1. Duoidal Categories. Duoidal categories result from the interaction of two monoidal categories. By the Eckmann-Hilton argument, each time we have two monoids $(*, \circ)$ such that one is a monoid homomorphism over the other, $(a \circ b) * (c \circ d) = (a * c) \circ (b * d)$, we know that both monoids coincide in a single commutative monoid.

However, an extra dimension helps us side-step the Eckmann-Hilton argument. If, instead of equalities or isomorphisms, we use directed morphisms, both monoids (which now may become 2-monoids) do not necessarily coincide, and the resulting structure is that of a duoidal category.

Definition 3.1.1 (Duoidal category)**.** A *duoidal category* [AM10] is a category $\mathbb{C}$ with two monoidal structures, $(\mathbb{C}, \otimes, I, \alpha, \lambda, \rho)$ and $(\mathbb{C}, \triangleleft, N, \beta, \kappa, \nu)$ such that the latter distribute over the former. In other words, it is endowed with a duoidal tensor, $(\triangleleft) \colon \mathbb{C} \times \mathbb{C} \to \mathbb{C}$, together with natural distributors

$$\begin{aligned} &\psi_2 \colon (X \triangleleft Z) \otimes (Y \triangleleft W) \to (X \otimes Y) \triangleleft (Z \otimes W), \\ &\psi_0 \colon I \to I \triangleleft I, \\ &\varphi_2 \colon N \otimes N \to N, \quad \text{and} \\ &\varphi_0 \colon I \to N, \end{aligned}$$

satisfying the following coherence equations (Appendix 0.1, Figures 1 to 5). A duoidal category is *strict* when both of its monoidal structures are.

Remark 3.1.2. In other words, the duoidal tensor and unit are lax monoidal functors for the first monoidal structure, which means that the laxators must satisfy the following equations.

(1) $(\psi_2 \otimes id) \fatsemi \psi_2 \fatsemi (\alpha \triangleleft \alpha) = \alpha \fatsemi (id \otimes \psi_2) \fatsemi \psi_2$, for the associator;
(2) $(\psi_0 \otimes id) \fatsemi \psi_2 \fatsemi (\lambda \triangleleft \lambda) = \lambda$, for the left unitor; and
(3) $(id \otimes \psi_0) \fatsemi \psi_2 \fatsemi (\rho \triangleleft \rho) = \rho$, for the right unitor;
(4) $\alpha \fatsemi (id \otimes \varphi_2) \fatsemi \varphi_2 = (\varphi_2 \otimes id) \fatsemi \varphi_2$, for the associator;
(5) $(\varphi_0 \otimes id) \fatsemi \varphi_2 = \lambda$, for the left unitor; and
(6) $(id \otimes \varphi_0) \fatsemi \varphi_2 = \rho$, for the right unitor.

3.1.2. Communication via Duoidals. The operations of a posetal duoidal structure can be interpreted as speaking about the communication of processes [SS22]. Let $(\otimes, i)$ and $(\triangleleft, n)$ form a duoidal structure on a poset. We read the elements of this poset as being processes and we interpret

(1) $x \otimes y$ as "x and y happen together, in parallel and independently";
(2) $x \triangleleft y$ as "y happens after x, and may depend on it";
(3) i as a process that "interrupts communication";
(4) n as a process that "does nothing";
(5) $x \to y$ as "channels of x are included in channels of y".

Under this interpretation, the rules of a duoidal category say that

(1) $(x \triangleleft y) \otimes (z \triangleleft w) \to (x \otimes z) \triangleleft (y \otimes w)$, adds intermediate communication;
(2) $i \to i \triangleleft i$, allows to interrupt an interrupted process;
(3) $n \otimes n \to n$, simplifies parallelism that does nothing; and
(4) $i \to n$, allows new communications.

We can already picture this interpretation in terms of *communication diagrams*. Processes are boxes, and wires represent the information flow: a path from a process to another one means that the first can communicate to the second (Figure 1).

The axioms of a duoidal category impose the following transformations of communication diagrams (Figure 2).

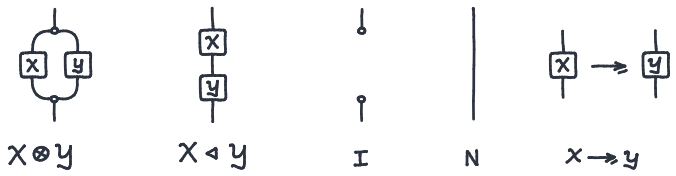


FIGURE 1. Communication diagrams.

We can take these diagrams seriously: they are string diagrams of a monoidal bicategory where processes are endocells. The only structure that we ask of the single object generator is that of an *adjoint monoid*.

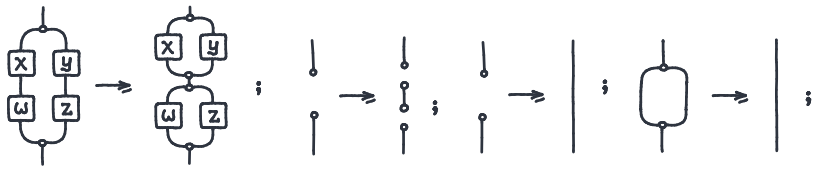


FIGURE 2. Communication diagram axioms.

3.1.3. Duoidals via adjoint monoids. Let $(\mathbb{C}, \otimes, I)$ be a monoidal category; let $(A, \curlyvee, \wr)$ be a monoid and let $(B, \curlywedge, \curlyvee)$ be a comonoid. The set of morphisms $\mathbb{C}(A; B)$ forms a monoid with the operation of *convolution*, $f * g = \curlywedge \fatsemi (f \otimes g) \fatsemi \curlyvee$ and the biunit, $e = \curlyvee \fatsemi \wr$, as in Figure 3.

FIGURE 3. Convolution monoid.

Convolution and composition interact duoidally whenever the monoid and the comonoid are adjoint to each other. This is the notion of an *adjoint monoid*.

Definition 3.1.3. An *adjoint monoid*, in a monoidal bicategory $\mathbb{B}$, is an object endowed with both 2-monoid and 2-comonoid structure $(A, \curlyvee, \wr, \curlyvee, \curlywedge)$, such that the multiplication is adjoint to the comultiplication $(\curlyvee \dashv \curlywedge)$ and the unit is adjoint to the counit $(\wr \dashv \curlyvee)$.

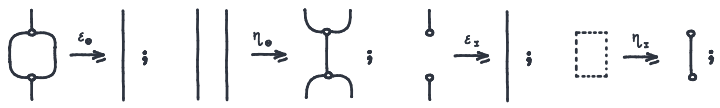


FIGURE 4. Adjoint monoid.

This means that there exist 2-cells, $\varepsilon_\otimes : \curlywedge \fatsemi \curlyvee \to \mathrm{id}$ and $\eta_\otimes : \mathrm{id} \to \curlywedge \fatsemi \curlyvee$, witnessing the adjunction $(\curlyvee \dashv \curlywedge)$; and that there exist 2-cells, $\varepsilon_I : \curlyvee \fatsemi \wr \to \mathrm{id}$ and $\eta_I : \mathrm{id} \to \curlyvee \fatsemi \wr$ witnessing the adjunction $(\wr \dashv \curlyvee)$, as in Figure 4.

THEOREM 3.1.4 (Garner, López Franco [GF16]). *The endocells of an adjoint 2-monoid form a duoidal category with convolution and composition.*

PROOF. Let $(A, \curlyvee, \wr, \curlywedge, \curlywedge)$ be an adjoint monoid, and let $X, Y \in \mathbb{B}(A; A)$ be endocells. We define the sequential tensor as the composition, $X \triangleleft Y = X \fatsemi Y$, with the unit being the identity, $N = \mathbf{I}$. We define the parallel tensor as the convolution, $X \otimes Y = \curlywedge \fatsemi (X \boxtimes Y) \fatsemi \curlyvee$, with the unit being the pair of monoid units, $I = \curlywedge \fatsemi \wr$. The duoidal interchangers are constructed out of the 2-cells of the adjoint monoid, taking Figure 2 seriously as the string diagrams of a monoidal bicategory.

(1) The first interchanger, $(X \triangleleft W) \otimes (Y \triangleleft Z) \to (X \otimes Y) \triangleleft (W \otimes Z)$, is constructed from the tensor adjunction unit, $\eta_{\otimes} \colon \mathrm{id} \to \curlyvee \fatsemi \curlywedge$;
(2) the second interchanger, $I \to I \triangleleft I$, is constructed from the unit of the unit adjunction, $\eta_I \colon \mathrm{id} \to \wr \fatsemi \curlywedge$;
(3) the third interchanger, $I \to N$, is constructed from the counit of the unit adjunction, $\varepsilon_I \colon \curlywedge \fatsemi \wr \to \mathrm{id}$; and
(4) the fourth interchanger, $N \otimes N \to N$, is constructed from the tensor adjunction counit, $\varepsilon_{\otimes} \colon \curlywedge \fatsemi \curlyvee \to \mathrm{id}$.

Finally, we need to check that all of the structure diagrams commute. This is usually left to the reader [GF16]; we can visualize the equations as surface diagrams. □

Remark 3.1.5 (Day convolution)**.** A particular case is the convolution of two parallel profunctors $P, Q \colon \mathbb{C} \to \mathbb{D}$ between monoidal categories. A monoidal category determines an adjoint monoid in the monoidal bicategory of profunctors. Convolution is the operation that constructs a profunctor $P * Q \colon \mathbb{C} \to \mathbb{D}$ defined by

$$(P * Q)(A, B) = \int^{X,X',Y,Y'} \mathbb{C}(A, X \otimes Y) \times P(X, X') \times Q(Y, Y') \times \mathbb{D}(X' \otimes Y', B).$$

Whenever we particularize to presheaves over a monoidal category $\mathbb{C}$, we recover *Day convolution* of presheaves.

$$(F * G)(A) = \int^{X,Y} \mathbb{C}(A, X \otimes Y) \times F(X) \times G(Y).$$

In particular, the endoprofunctors over any monoidal category form a duoidal category, this is the duoidal category that we study in Section 3.3.

3.1.4. Be Careful with Duoidal Coherence. Monoidal categories possess a coherence theorem that determines that any two parallel morphisms constructed out of structure isomorphisms commute. In contrast, duoidal categories do not satisfy that same statement. This causes some confusion around coherence for duoidal categories. I bring an example of how this confusion may arise, hoping that it will help the interested reader and that it may further justify the importance of expository category theory.

We could be tempted to provide an alternative definition of duoidal categories that avoids asking for a bunch of commutative diagrams by simply asking that any formal such diagram commutes. In fact, this may possible for the physical duoidal categories studied by Spivak and Shapiro [SS22], who comment that

> *alternatively, duoidal categories can be defined by the two monoidal structures along with the generating structure maps [...] (4) natural in a, b, c, d which satisfy equations guaranteeing that any two structure maps built from those in (4) between the same two expressions in $y_{\otimes}, y_{\triangleleft}, \otimes, \triangleleft$ are equal.*

One of the first, most complete and comprehensive accounts of duoidal categories is the monograph by Aguiar and Mahajan [AM10]. It includes a passing comment that could suggest that this version can be proven correct. It says that

> *"[...] if two morphisms $A \to B$ are constructed out of the structure maps in $\mathbb{C}$ (including the structure constraints of the monoidal categories $(\mathbb{C}, \diamond, I)$ and $(\mathbb{C}, \star, J)$), then they coincide."*

However, intepreted literally and strictly, this turns out to not be true. Two parallel morphisms constructed out of the structure maps of a duoidal category do not need to coincide.

Proposition 3.1.6. *There exist two different maps of type $I \triangleleft I \to I$ constructed out of the structure maps of a duoidal category.*

Proof. We can consider two maps of type $I \triangleleft I \to I$, depending on which of the two parallel units we decide to convert to a sequential unit using the laxators. Explicitly, we are saying that $(I \triangleleft \varphi_0) \fatsemi \rho_\triangleleft$ and $(\varphi_0 \triangleleft I) \fatsemi \lambda_\triangleleft$ do not coincide; and the more intuitive string diagrams for bicategories for adjoint monoids confirm this (Figure 5). We will construct an explicit example of this phenomenon.

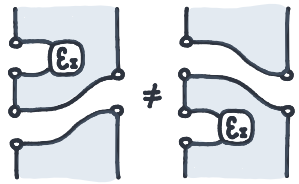


Figure 5. Bicategorical string diagrams for the two coherence maps.

Consider the duoidal category of endoprofunctors over a monoidal category. This is one of the first examples of duoidal category described by Street [Str12]; it is also described by Garner and López Franco [GF16], even when the axioms are not explicitly checked in print. In this category of endoprofunctors over $\mathbb{C}$, parallel tensor is the profunctor $I(X;Y) = \mathbb{C}(X;I) \times \mathbb{C}(I;Y)$, and sequencing two of them gives

$$(I \triangleleft I)(X;Y) = \hom(X;I) \times \hom(I;I) \times \hom(I;Y).$$

In this case, the two maps send the triple (f, a, g) to $(f \fatsemi a, g)$ and $(f, a \fatsemi g)$, respectively. However, these two pairs do not need to be equal if $a \in \hom(I;I)$ is a non-identity morphism. □

Example 3.1.7 (Graded spaces)**.** We look for a more classical source of examples in the theory of *graded spaces*. Let $(\mathbb{V}, \otimes, I)$ be a monoidal category with coproducts that are preserved by the tensor; let $(G, +, 0)$ be a commutative monoid. We say that the functor category $[G, \mathbb{V}]$ is the category of *G-graded $\mathbb{V}$-spaces*. This category has a rich structure; we highlight two of its tensor products: the *pointwise* or *Hadamard* tensor product

$$(V \otimes W)_n = V_n \otimes W_n, \text{ for each } n \in G, \text{ with unit } \mathbb{I}_n = I;$$

and the *convolution* or *Cauchy* tensor product

$$(V \bullet W)_n = \sum\nolimits_{k+m=n} V_k \otimes W_m, \text{ with unit } \mathbf{1}_n = \mathbf{0} \text{ except for } \mathbf{1}_0 = I.$$

These two tensors interact in a duoidal category with a laxator as follows; see for instance the work of López Franco and Vasilakopoulou [FV20].

$$\sum_{k+m=n} V_k \otimes W_k \otimes U_m \otimes Z_m \to \left(\sum_{k_1+m_1=n_1} V_{k_1} \otimes U_{m_1} \right) \otimes \left(\sum_{k_2+m_2=n_2} W_{k_2} \otimes Z_{m_2} \right).$$

Proposition 3.1.8. *Dually, there exist two different maps of type $J \to J \otimes J$ constructed out of the structure maps of a duoidal category.*

PROOF. This follows from the previous Proposition 3.1.6, by considering the opposite duoidal category. However, let us comment a second example [AM]. Consider the duoidal category of graded spaces over a monoid G. The two maps, $\mathbb{I} \to \mathbb{I} \otimes 1 \to \mathbb{I} \otimes \mathbb{I}$ and $\mathbb{I} \to 1 \otimes \mathbb{I} \to \mathbb{I} \otimes \mathbb{I}$, correspond to inclusions of the vector space graded by $g \in G$ into the summand indexed by $(g, 0)$ or $(0, g)$, respectively; these are different in general. □

In fact, the stronger statement of coherence does not seem to be used explicitly in any of these two texts, and the definition of duoidal categories as completely coherent strucutres is not usually found in the literature. Most authors, like Aguiar and Mahajan [AM10], and Garner and López Franco [GF16], revert to the definition of duoidal category as a 2-monoid in the monoidal bicategory of monoidal categories.

Aguiar and Mahajan [AM10] do actually point out that the expected coherence theorem should follow from the coherence theorem for lax monoidal functors. The confusion can arise if we do not realize that this coherence theorem does not actually prove that any two parallel maps coincide: in particular, coherence for lax monoidal functors does not prove that the two maps $F(I) \otimes F(I) \to F(I)$ coincide. In this case, however, the problem is better known – it is mentioned by Malkiewich and Ponto [MP21], who cite a short mention in the original proof by Lewis [Lew06] and Kelly and Laplaza [KL80].

3.1.5. Bibliography. I thank Marcelo Aguiar and Swapneel Mahajan [AM] for their generosity, helping me confirm the problem and specially for providing the second counterexample; I thank Brandon Shapiro and David Spivak for helping me follow this same idea on their work. I thank Matt Earnshaw for sharing his knowledge of the literature on duoidal categories.

3.2. Normal Duoidal Categories

Duoidal categories seem to contain too much structure: of course, we want to split things in two different ways, sequentially ($\lhd$) and in parallel ($\otimes$); but that does not necessarily mean that we want to keep track of two different types of units, parallel (I) and sequential (N). The atomic components of our decomposition algebra could be the same, without having to care if they are *atomic for sequential composition* or *atomic for parallel composition*; when this is the case, we talk of *normal duoidal categories*.

Definition 3.2.1. A *normal duoidal category* is a duoidal category in which the map $\varphi_0 \colon I \to N$ is an isomorphism.

While duoidal categories are useful to track communication between processes; symmetric normal duoidal categories track *dependencies* – whether a process' input depends on the output of another – structuring a dependency poset. This idea is explored by Garner and López Franco [GF16] and Spivak and Shapiro [SS22], and it has a formal counterpart in Theorem 3.2.9.

Most duoidal categories we have seen so far – and particularly those arising from adjoint monoids – have two different units. There exists a well-known abstract procedure that, starting from some duoidal category, constructs a new duoidal category that is normal: both units are identified. This procedure is known as *normalization*, and it can only be applied to duoidal categories with certain coequalizers preserved by the tensor.

3.2.1. Normalization of duoidal categories. Garner and López Franco construct the normalization of a well-behaved duoidal category, using a new duoidal category of *bimodules* [GF16].

Remark 3.2.2. Let M be a bimonoid in the duoidal category $(\mathbb{V}, \otimes, I, \lhd, N)$, with maps $e \colon I \to M$ and $m \colon M \otimes M \to M$; and with maps $u \colon M \to N$ and $d \colon M \to M \lhd M$. Consider now the category of $M^{\otimes}$-bimodules. This category has a monoidal structure lifted from $(\mathbb{V}, \lhd, N)$:

(1) the unit, N, has a bimodule structure with

$$M \otimes N \otimes M \overset{u \otimes \mathrm{id} \otimes u}{\longrightarrow} N \otimes N \otimes N \longrightarrow N;$$

(2) the sequencing of two $M^{\otimes}$-bimodules is a $M^{\otimes}$-bimodule with

$$\begin{aligned} &M \otimes (A \lhd B) \otimes M \\ &\quad \to (M \lhd M) \otimes (A \lhd B) \otimes (M \lhd M) \\ &\quad \to (M \otimes A \otimes M) \lhd (M \otimes B \otimes M) \to A \lhd B. \end{aligned}$$

Moreover, whenever $\mathbb{V}$ admits reflexive coequalizers preserved by $(\otimes)$, the category of $M^{\otimes}$-bimodules is monoidal with the tensor of bimodules: the coequalizer

$$A \otimes M \otimes B \rightrightarrows A \otimes B \twoheadrightarrow A \otimes_M B.$$

In this case $(\mathbf{Bimod}^{\otimes}_M, \otimes_M, M, \lhd, N)$ is a duoidal category.

Theorem 3.2.3 (Normalization of a duoidal, [GF16]). *Let $(\mathbb{V}, \otimes, I, \lhd, N)$ be a duoidal category with reflexive coequalizers preserved by $(\otimes)$. The category of N-bimodules is then a normal duoidal category,*

$$\mathcal{N}(\mathbb{V}) = (\mathbf{Bimod}^{\otimes}_N, \otimes_N, N, \lhd, N).$$

We call this category the normalization *of the duoidal category $\mathbb{V}$.*

3.2.2. Physical duoidal categories. The interaction of dependent and independent composition of normal duoidal categories is a recurrent idea in physical models: categorical models of spacetime exhibit this structure [HK22, SS22]; but it is also exhibited by parallel and sequentially composing programs [HS23]; or more simply, by the category of partially ordered sets.

In most of these cases, the normal duoidal category has an extra property: the parallel tensor ($\otimes$) is symmetric. This is what motivates the name *physical duoidal category* for the $\otimes$-symmetric normal duoidal categories.

Definition 3.2.4. A *physical duoidal category* is a normal duoidal category endowed with a symmetric monoidal category structure for its parallel tensor.

Posets are a canonical example of a physical duoidal category: in fact, it is known that a subcategory of the category of posets and poset inclusions forms the free physical duoidal category over a generator. In that precise sense, duoidal expressions are dependency tracking posets.

Definition 3.2.5. The category of *poset shapes*, **PosetSh**, is the skeleton of the category of finite posets with bijective-on-objects monotone functions. Objects are isomorphism classes of finite posets, and morphisms are inclusions.

Proposition 3.2.6. Poset shapes *form a physical duoidal category.*

Proof. The sequential tensor is constructed by sequentially joining the posets. Let $(P, \leq_P)$ and $(Q, \leq_Q)$ be two posets; their sequentiation, $P \triangleleft Q$, is a poset that contains a copy of P, a copy of Q, and an edge $p_i \leq q_j$ for each $p_i \in P$ and $q_j \in Q$; that is,

$$P \triangleleft Q = (P + Q, \leq_P + \leq_Q + \{p_i \leq q_j \mid p_i \in P, q_j \in Q\}).$$

The parallel tensor, $P \otimes Q$, is defined to be the disjoint union of posets, $P \otimes Q = (P + Q, \leq_P + \leq_Q)$, which defines a symmetric monoidal structure. The empty poset is the unit for both sequential and parallel tensoring, making **PosetSh** a physical duoidal category. □

The category of poset shapes is not posetal: there are, for instance, two possible inclusions of the discrete two-element poset into itself. This prompts us to label the nodes to indicate inclusions, as in Figure 6, but we work up to relabelling, or α-equivalence.

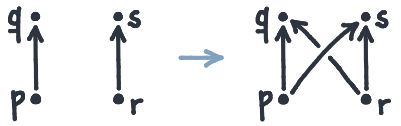


Figure 6. Poset inclusion.

What makes this physical duoidal category particularly relevant is that it contains the free physical duoidal category over a generator. Every formal normal duoidal expression constructs a poset: we simply substitute each variable by the singleton poset and we interpret the expression in the duoidal category of poset shapes. Every formal structure map between normal duoidal expressions corresponds to an inclusion; for instance, Figure 6 documents the structure map $(p \triangleleft q) \otimes (r \triangleleft s) \to (p \otimes r) \triangleleft (q \otimes s)$.

Formalizing this result needs a bit of care, though: while all formal physical duoidal expressions correspond to posets, not every poset shape corresponds to a physical duoidal expression. The poset shapes that arise from applying duoidal operations to the singleton poset are called *expressible*, and we have a characterization result for them.

Definition 3.2.7. Expressible poset shapes are those inductively constructed from

(1) the empty poset, N;
(2) the singleton poset, $\{A\}$;
(3) the union of posets, $P \otimes Q = (P + Q, \leq_P + \leq_Q)$; and
(4) the sequencing of posets,

$$P \triangleleft Q = (P + Q, \leq_P + \leq_Q + \{p_i \leq q_j \mid p_i \in P, q_j \in Q\}).$$

Expressible poset shapes form a full duoidal subcategory of the physical duoidal category of poset shapes, **ExprSh**.

Proposition 3.2.8 (Grabowski, [Gra81])**.** *Not every poset shape is expressible. In fact, a poset shape is not expressible if and only if it admits an inclusion of the* Z-poset shape *defined by Figure 7.*

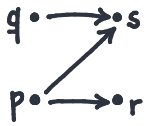


FIGURE 7. The Z poset shape.

THEOREM 3.2.9 (Shapiro and Spivak, [SS22]). *The physical duoidal category* **ExprSh** *of expressible poset shapes is the free physical duoidal category on a single object. There exists exactly one structure map between any two objects of the free physical duoidal category for each inclusion of their associated expressible posets.*

Remark 3.2.10 (Coherence for normal duoidal categories)**.** Coherence for duoidal categories needs some care: not any two morphisms between distinctly-typed expressions in the free duoidal category are equal (Proposition 3.1.6). However, the previous theorem implies that the same statement is true for normal duoidal categories.

Corollary 3.2.11. *Any two morphisms between distinctly typed expressions in the free duoidal category over a set of objects are equal.*

3.2.3. Physical Lax Tensor of a Physical Duoidal Category. Let us recap our interpretation of physical duoidal categories: they track an underlying poset of dependencies. The sequential tensor, $X \triangleleft Y$, says that X occurs before Y, but $Y \triangleleft X$ says that Y occurs before X; consequently, it is not symmetric. The parallel tensor, $X \otimes Y$, states that both X and Y occur independently. This final section of our introduction to physical duoidal categories shows what happens when we want to consider both X and Y but we do not know at all how they interact: the tensor that tracks this case is a derived operation, the *physical tensor*, $X \boxtimes Y$.

The physical tensor simply says that both occur at some point: it does not impose independence, but it does not impose any particular dependency either. The physical tensor $X \boxtimes Y$ says that X may occur before Y, or Y before X, or both in parallel and in that case it does not matter how we regard the dependency. This is a tool that we will employ later to discuss a version of monoidal context that does not track dependency: *wiring diagrams* ([Spi13], Conjecture 3.5.11).

Remark 3.2.12. The binary physical tensor, $X \boxtimes Y$, is easy to define: it is the pushout of the two structure maps $X \otimes Y \to X \triangleleft Y$ and $X \otimes Y \to Y \triangleleft X$. However, unlike most tensors, its n-ary version cannot be deduced from its binary and nullary versions; the physical tensor is only a lax tensor.

Definition 3.2.13 (Leinster [Lei04])**.** A *lax monoidal category* is a category $\mathbb{C}$ endowed with a family of lax tensor n-fold tensor functors $(\boxtimes)\colon \mathbb{C}^n \to \mathbb{C}$ – written as $X_1 \boxtimes \ldots \boxtimes X_n$, with the 0-ary case E – and a family of associator natural transformations that unbias the application of the lax tensor,

$$\alpha\colon \boxtimes_{i=0}^{n} \left(\boxtimes_{j=0}^{k_i} X_j^i\right) \to X_1^1 \boxtimes \ldots \boxtimes X_{k_1}^1 \boxtimes \ldots \boxtimes X_1^n \boxtimes \ldots \boxtimes X_{k_n}^n,$$

such that all formally well-typed equations hold.

Definition 3.2.14. Let $(\mathbb{C}, \otimes, I, \triangleleft, N)$ be a physical duoidal category. The *physical tensor*, $(\boxtimes)$, is an additional lax monoidal tensor, defined as the glueing of the sequential tensor $(\triangleleft)$ along the parallel tensor $(\otimes)$; that is, it is the pushout on the following family of structure maps, indexed by permutations

$$l_\sigma\colon X_1 \otimes \ldots \otimes X_n \to X_{\sigma 1} \triangleleft \ldots \triangleleft X_{\sigma n}, \text{ for } \sigma \in P(n).$$

Remark 3.2.15. This only forms a lax tensor for a good reason. Consider the simpler case of three elements, $X \boxtimes Y \boxtimes Z$. This expression allows all of the possible six permutations to occur: *(i)* $X \triangleleft Y \triangleleft Z$; *(ii)* $X \triangleleft Z \triangleleft Y$; *(iii)* $Y \triangleleft X \triangleleft Z$; *(iv)* $Y \triangleleft Z \triangleleft X$; *(v)* $Z \triangleleft X \triangleleft Y$; and *(vi)* $Z \triangleleft Y \triangleleft X$. However, when we consider $(X \boxtimes Y) \boxtimes Z$, we are only allowing a certain subset of these cases to occur. Namely, only those where Z does not happen between X and Y: *(i)* $X \triangleleft Y \triangleleft Z$; *(ii)* $Y \triangleleft X \triangleleft Z$; *(iii)* $Z \triangleleft X \triangleleft Y$; and *(iv)* $Z \triangleleft Y \triangleleft X$. This is why the physical tensor is only lax. We have two inclusions: $(X \boxtimes Y) \boxtimes Z \to X \boxtimes Y \boxtimes Z$ and $X \boxtimes (Y \boxtimes Z) \to X \boxtimes Y \boxtimes Z$, but these are not isomorphisms.

Proposition 3.2.16. *The physical tensor defines a lax monoidal structure. Given any physical duoidal category* $(\mathbb{C}, \otimes, \triangleleft, N)$*, the physical tensor defines a lax monoidal category* $(\mathbb{C}, \boxtimes, N)$*.*

Proof. The definition of the laxator follows from the universal property of the pushout; the coherence equations hold by uniqueness of the maps constructed out of this universal property. □

3.2.4. Bibliography. The original monograph on duoidal categories is due to Aguiar and Mahajan [AM10] – duoidal categories were originally known as *"2-monoidal categories"*; Street first described multiple examples that we recall [Str12], and Garner and López Franco mention for the first time the connection to adjoint monoids [GF16]. The reason duoidal categories do not have a correspondence in lower dimensional algebra is the Eckmann-Hilton argument [EH61, Theorem 1.12].

Physical duoidal categories follow the definition and nomenclature of Spivak and Shapiro [SS22]; their work makes the case for interpreting them as expressing dependencies between processes and argues initiality of the category of expressible posets. It seems originally due to Grabowski [Gra81] that expressible posets are precisely those not containing a Z, and Gischer recognized the lax interchange of normal duoidal categories as subsumption of posets [Gis88]. Even if the physical lax tensor does not seem to appear in the related literature, its definition and its consideration as a lax tensor follow the work of Leinster [Lei04] and the abstraction of commutativity by Garner and López Franco [GF16]. I thank Matt Earnshaw for multiple pointers to the literature.

3.3. Produoidal Decomposition of Monoidal Categories

3.3.1. Produoidal categories. Produoidal categories, first defined by Booker and Street [BS13], provide an algebraic structure for the interaction of sequential and parallel decomposition. A produoidal category $\mathbb{V}$ not only contains *morphisms*, $\mathbb{V}(X;Y)$, as in a category, but also *sequential joints*, $\mathbb{V}(X_0 \lhd X_1;Y)$, and *sequential units*, $\mathbb{V}(N;X)$, provided by a promonoidal structure; and *parallel joints*, $\mathbb{V}(X_0 \otimes X_1;Y)$ and *parallel units*, $\mathbb{V}(I;X)$, provided by another promonoidal structure.

These splits must be coherent. For instance, imagine we want to join X_0, X_1 and X_2 (sequentially) into Y. Joining X_0 and X_1 into something $(\bullet)$, and then joining that something with X_2 to produce Y *should be doable in essentially the same ways* as joining X_1 and X_2 into something $(\bullet)$, and then joining that something with X_0 to produce Y. Formally, we are saying that

$$\mathbb{V}(X_0 \lhd X_1;\bullet) \diamond \mathbb{V}(\bullet \lhd X_2;Y) \cong \mathbb{V}(X_1 \lhd X_2;\bullet) \diamond \mathbb{V}(X_0 \lhd \bullet;Y),$$

and, in fact, we just write $\mathbb{V}(X_0 \lhd X_1 \lhd X_2;Y)$ for the set of such transformations. This is precisely what we ask for in a promonoidal structure.

Definition 3.3.1 (Produoidal category)**.** A *produoidal category* is a category $\mathbb{V}$ endowed with two promonoidal structures,

$$\mathbb{V}(\bullet \otimes \bullet;\bullet)\colon \mathbb{V}\times\mathbb{V}\to\mathbb{V}, \text{ and } \mathbb{V}(I;\bullet)\colon 1\to\mathbb{V},$$
$$\mathbb{V}(\bullet \lhd \bullet;\bullet)\colon \mathbb{V}\times\mathbb{V}\to\mathbb{V}, \text{ and } \mathbb{V}(N;\bullet)\colon 1\to\mathbb{V},$$

such that one laxly distributes over the other. This is to say that it is endowed with the following natural *lax interchangers*:

(1) $\psi_2\colon \mathbb{V}((X \lhd Y)\otimes(Z \lhd W);\bullet) \to \mathbb{V}((X\otimes Z)\lhd(Y\otimes W);\bullet)$,
(2) $\psi_0\colon \mathbb{V}(I;\bullet) \to \mathbb{V}(I \lhd I;\bullet)$,
(3) $\varphi_2\colon \mathbb{V}(N\otimes N;\bullet) \to \mathbb{V}(N;\bullet)$, and
(4) $\varphi_0\colon \mathbb{V}(I;\bullet) \to \mathbb{V}(N;\bullet)$.

Interchangers, together with unitors and associators, must satisfy coherence conditions (see Appendix 0.1). We denote by **ProDuo** the category of produoidal categories and produoidal functors.

Proposition 3.3.2. *Let $\mathbb{V}$ be a produoidal category, then its category of copresheaves, $[\mathbb{V}, \mathbf{Set}]$, is a duoidal category.*

Remark 3.3.3 (Nesting profunctorial structures)**.** Notation for nesting functorial structures, say $(\lhd)$ and $(\otimes)$, is straightforward: we use expressions like $(X_1 \otimes Y_1) \lhd (X_2 \otimes Y_2)$ without a second thought. Nesting the profunctorial (or *virtual*) structures $(\lhd)$ and $(\otimes)$ is more subtle: defining $\mathbb{V}(X\otimes Y;\bullet)$ and $\mathbb{V}(X\lhd Y;\bullet)$ for each pair of objects X and Y does not itself define what something like $\mathbb{V}((X_1\otimes Y_1)\lhd(X_2\otimes Y_2);\bullet)$ means. Recall that, in the profunctorial case, $X_1 \lhd Y_1$ and $X_1 \otimes Y_1$ are not objects themselves: they are just names for the profunctors $\mathbb{V}(X_1 \lhd Y_1;\bullet)$ and $\mathbb{V}(X_1\otimes Y_1;\bullet)$, which are not *representable*.

Instead, when we write $\mathbb{V}((X_1 \otimes Y_1) \lhd (X_2 \otimes Y_2);\bullet)$, we formally mean the composition of profunctors $\mathbb{V}(X_1\otimes Y_1;\bullet_1)\diamond\mathbb{V}(X_2\otimes Y_2;\bullet_2)\diamond\mathbb{V}(\bullet_1\lhd\bullet_2;\bullet)$. By convention, nesting profunctorial structures means profunctor composition in this text.

Remark 3.3.4. Should we reverse the direction of the interchangers? Depending on the author, promonoidal categories and produoidal categories are reversed. It seems that both conventions have their advantages. The one we follow here [DPS05] makes intuitive sense: it follows the multicategorical and operadic point of view – multiple ingredients produce a result. The opposite one [EHR23] gets the

interchangers to be those of a duoidal category and it becomes clear that there is a correspondence between produoidal categories and closed duoidal categories on preshaves.

3.3.2. Monoidal Contour of a Produoidal Category. Any produoidal category freely generates a monoidal category, its *monoidal contour*. Contours form a monoidal category of paths around the decomposition trees of the produoidal category. Contours follow a pleasant geometric pattern where we follow the shape of the decomposition, both in the parallel and sequential dimensions, to construct both sequential and parallel compositions for a monoidal category.

Definition 3.3.5 (Monoidal contour)**.** The *contour* of a produoidal category $\mathbb{B}$ is the monoidal category $\mathsf{mContour}(\mathbb{B})$ presented by two polarized objects, X° and $X^\bullet$, for each object $X \in \mathbb{B}_{\text{obj}}$; and generated by arrows that arise from *contouring* both sequential and parallel decompositions of the promonoidal category.

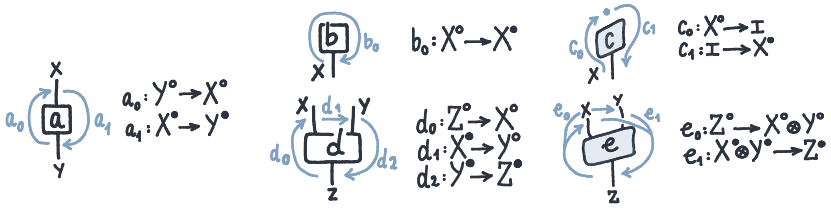


FIGURE 8. Generators of the monoidal category of contours.

Specifically, monoidal contour is the monoidal category presented by the following generators in Figure 8:

(1) $a_0 \colon Y^\circ \to X^\circ$ and $a_1 \colon X^\bullet \to Y^\bullet$, for each morphism $a \colon X \to Y$;
(2) $b_0 \colon X^\circ \to X^\bullet$, for each sequential unit, $b \colon \mathsf{N} \to X$;
(3) $c_0 \colon X^\circ \to I$ and $c_1 \colon I \to X^\bullet$, for each parallel unit, $c \colon I \to X$;
(4) a triple of generators $d_0 \colon Z^\circ \to X^\circ$, $d_1 \colon X^\bullet \to Y^\circ$ and $d_2 \colon Y^\bullet \to Z^\bullet$, for each sequential join $d \colon X \triangleleft Y \to Z$; and
(5) a pair of generators $e_0 \colon Z^\circ \to X^\circ \otimes Y^\circ$ and $e_1 \colon X^\bullet \otimes Y^\bullet \to Z^\bullet$ for each parallel join, $e \colon X \otimes Y \to Z$.

We impose all the equations of the theory of contour. Additionally, we also impose all of the equations depicted in Figure 9. Together, these form the theory of monoidal contour, which adds to the theory of sequential contour a new monoidal dimension.

Proposition 3.3.6. *Monoidal contour extends to a functor*

$$\mathsf{mContour} : \mathbf{ProDuo} \to \mathbf{MonCat}.$$

PROOF. Definition 3.3.5 defines how the functor acts on objects. We define the action on produoidal functors, the morphisms of the category of produoidal categories. Given a produoidal functor, $F : \mathbb{V} \to \mathbb{W}$, let us define the strict monoidal functor $\mathsf{mContour}(F) : \mathsf{mContour}(\mathbb{V}) \to \mathsf{mContour}(\mathbb{W})$ by the following morphism of generators:

(1) objects X° and $X^\bullet$ are mapped to $F(X)^\circ$ and $F(X)^\bullet$;
(2) for each $a \colon X \to Y$, the morphisms $a_0 : X^\circ \to X^\circ, a_1 : X^\bullet \to Y^\bullet$ are mapped to $F(a)_0$ and $F(a)_1$;
(3) for each $b \colon I \to X$, both $b_0 : X^\circ \to I$ and $b_1 : I \to X^\bullet$ are mapped to $F_I(b)_0$ and $F_I(b)_1$;
(4) for each $c \colon \mathsf{N} \to X$, the morphism $c_0 : X^\circ \to X^\bullet$ is mapped to $F_{\mathsf{N}}(c)_0$;

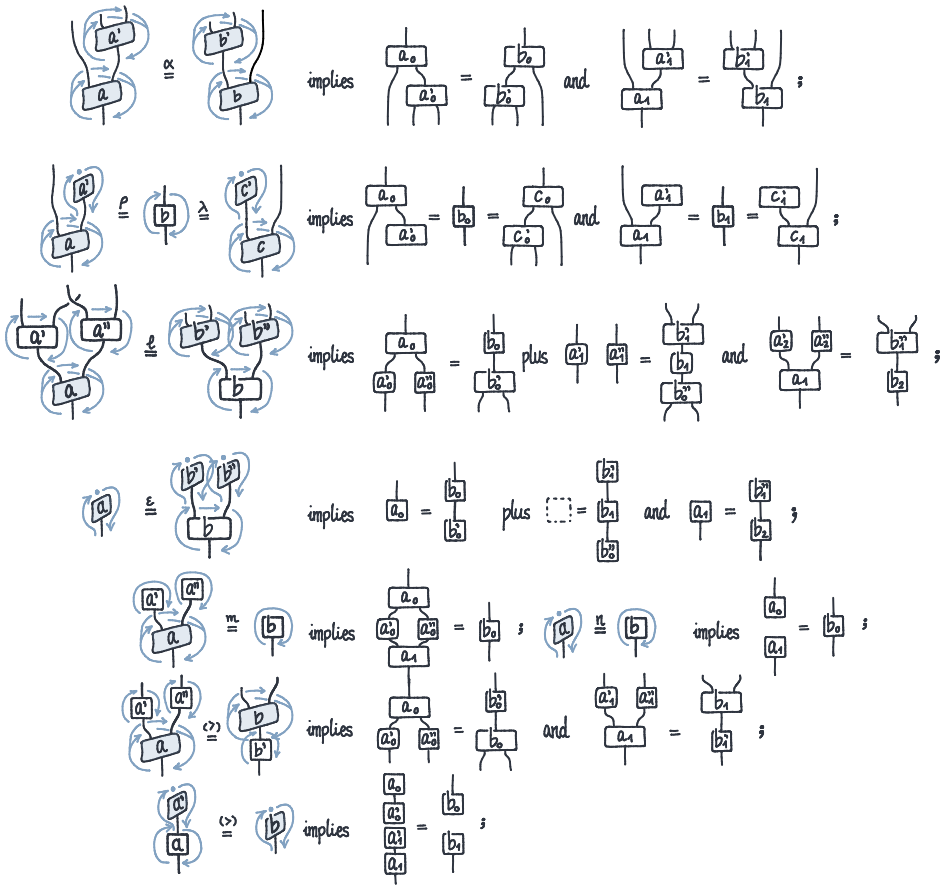


FIGURE 9. Extra equations for the theory of monoidal contour.

(5) for each $d\colon X \triangleleft Y \to Z$, the morphisms $d_0 : Z^\circ \to X^\circ$, $d_1 : X^\bullet \to Y^\circ$ and $d_2 : Y^\bullet \to Z^\bullet$ are mapped to $F_\triangleleft(d)_0$, $F_\triangleleft(d)_1$ and $F_\triangleleft(d)_2$;
(6) for each $e\colon X \triangleleft Y \to Z$, the morphisms $e_0 : Z^\circ \to X^\circ \otimes Y^\circ$, and $e_1 : X^\bullet \otimes Y^\bullet \to Z^\bullet$ are mapped to $F_\triangleleft(e)_0$, and $F_\triangleleft(e)_1$.

To show that this defines a morphism of presentations, we need to prove that the assignment of generators preserves the equations of the theory of contour, in Definition 2.4.1. Because $F : \mathbb{V} \to \mathbb{W}$ is a produoidal functor, the images of the generators do satisfy all of the contour equations of the target category. As a consequence, this assignment extends to a strict monoidal functor.

Finally, when $\mathrm{id}_\mathbb{V} : \mathbb{V} \to \mathbb{V}$ is an identity, the resulting functor is an identity because it is the identity on generators. Let $G : \mathbb{U} \to \mathbb{V}$ be another produoidal functor, then $\mathsf{mContour}(G \fatsemi F) = \mathsf{mContour}(G) \fatsemi \mathsf{mContour}(F)$ follows from the composition of produoidal functors. □

3.3.3. Produoidal Splice of a Monoidal Category. We want to go the other way around: given a monoidal category, what is the produoidal category that tracks the decomposition of arrows in that monoidal category? This subsection finds a right adjoint to the monoidal contour construction: the produoidal category of *spliced monoidal arrows*.

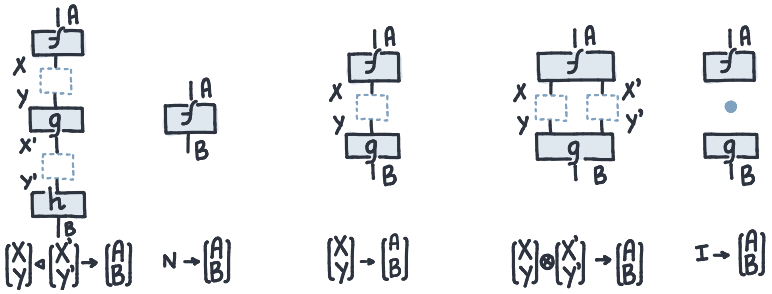


FIGURE 10. Spliced monoidal arrows.

Definition 3.3.7. Let $(\mathbb{C}, \otimes, I)$ be a monoidal category. Its produoidal category of *spliced monoidal arrows*, $\mathsf{mSplice}(\mathbb{C})$, has objects formed by pairs, $\mathsf{mSplice}(\mathbb{C})_{obj} = (\mathbb{C}^{op} \times \mathbb{C})_{obj}$, and is defined by the following profunctors, depicted in Figure 10.

(1) $\mathsf{mSplice}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \mathbb{C}(A; X) \times \mathbb{C}(Y, B)$,
(2) $\mathsf{mSplice}(\mathbb{C})(\begin{smallmatrix} X \\ Y \end{smallmatrix} \lhd \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; X) \times \mathbb{C}(Y; X') \times \mathbb{C}(Y'; B)$;
(3) $\mathsf{mSplice}(\mathbb{C})(\begin{smallmatrix} X \\ Y \end{smallmatrix} \otimes \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; X \otimes X') \times \mathbb{C}(Y \otimes Y'; B)$;
(4) $\mathsf{mSplice}(\mathbb{C})(\mathsf{N}; \begin{smallmatrix} A \\ B \end{smallmatrix};) = \mathbb{C}(A; B)$;
(5) $\mathsf{mSplice}(\mathbb{C})(I; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; I) \times \mathbb{C}(I; B)$.

Proposition 3.3.8. *Spliced monoidal arrows indeed form a produoidal category.*

PROOF SKETCH. The complete proof constructs all of the necessary natural isomorphisms using coend calculus. We refer to joint work of this author with Earnshaw and Hefford, where the equations are proven in full detail [EHR23]. □

Remark 3.3.9. The produoidal algebra of spliced arrows is a natural construction: abstractly, we know that there exists a duoidal structure on the endomodules of any monoidal category [Day70, Str12] – monoidal spliced arrows form its explicitly constructed produoidal counterpart. What may be more surprising is that spliced arrows have themselves a universal property as part of an adjunction.

THEOREM 3.3.10. *Spliced monoidal arrows form a produoidal category with their sequential and parallel splits, units, and suitable coherence morphisms and laxators. Spliced monoidal arrows extend to a functor* $\mathsf{mSplice} : \mathbf{MonCat} \to \mathbf{ProDuo}$. *The monoidal contour and the produoidal splice are left and right adjoints to each other, respectively.*

PROOF. Monoidal contour $\mathsf{mContour}(\mathbb{B})$ is presented by generators and equations: to specify a strict monoidal functor $\mathsf{mContour}(\mathbb{B}) \to \mathbb{M}$, it is enough to specify images of the generators and then prove that they satisfy the equations.

Let $(\mathbb{M}, \otimes_M, I_M)$ be a monoidal category. Then a strict monoidal functor $\mathsf{mContour}(\mathbb{B}) \to \mathbb{M}$ amounts to the following data satisfying some extra conditions.

(1) For each object $X \in \mathbb{B}_{\mathrm{obj}}$, a pair of objects $X^\circ, X^\bullet \in \mathbb{M}_{\mathrm{obj}}$;
(2) for each element $f\colon \mathsf{N} \to X$, a morphism $f_0\colon X^\circ \to X^\bullet$;
(3) for each unit $f\colon I \to X$, a choice of $f_0\colon X^\circ \to I$ and $f_1\colon I \to X^\bullet$;
(4) for each morphism $f\colon X \to Y$, a choice of $f_0\colon Y^\circ \to X^\circ$ and $f_1\colon X^\bullet \to Y^\bullet$;
(5) for each sequential split $f\colon X \lhd Y \to Z$, a choice of morphisms $f_0\colon Z^\circ \to X^\circ$, plus $f_1\colon X^\bullet \to Y^\circ$ and $f_2\colon Y^\bullet \to Z^\bullet$;
(6) for each parallel split $f\colon X \otimes Y \to Z$, a choice of morphisms $f_0\colon Z^\circ \to X^\circ \otimes Y^\circ$ and $f_1\colon X^\bullet \otimes Y^\bullet \to Z^\bullet$.

In order to construct a well-defined strict monoidal functor, the previous assignments must satisfy the following conditions for each one of the two promonoidal categories:

(1) $\alpha(a \fatsemi_1 b) = (c \fatsemi_2 d)$ in the promonoidal category implies $a_0 \fatsemi (b_0 \otimes \mathrm{id}) = c_0 \fatsemi (\mathrm{id} \otimes d_0)$ and $(b_1 \otimes \mathrm{id}) \fatsemi a_1 = (\mathrm{id} \otimes d_1) \fatsemi c_1$ in the monoidal category;
(2) $\lambda(a \fatsemi_1 b) = c = \rho(d \fatsemi_2 e)$ in the promonoidal category implies $a_0 \fatsemi (b_0 \otimes \mathrm{id}) = c_0 = d_0 \fatsemi (\mathrm{id} \otimes e_0)$ and $(b_1 \otimes \mathrm{id}) \fatsemi a_1 = c_1 = (\mathrm{id} \otimes e_1) \fatsemi d_1$ in the monoidal category;

Moreover, they must also satisfy the following conditions for the produoidal category.

(1) $\psi_2(a \mid b \mid c) = (d \mid e \mid f)$ in the promonoidal category implies $a_0 \fatsemi (b_0 \otimes c_0) = d_0 \fatsemi e_0, b_1 \otimes c_1 = e_1 \fatsemi d_1 \fatsemi f_0$ and $(b_2 \otimes c_2) \fatsemi a_1 = f_1 \fatsemi d_2$ in the monoidal category;
(2) $\psi_0(a) = (b \mid c \mid d)$ in the promonoidal category implies $a_0 = b_0 \fatsemi c_0$, $\mathrm{id} = c_1 \fatsemi b_1 \fatsemi d_0$, and $a_1 = d_1 \fatsemi b_2$ in the monoidal category;
(3) $\varphi_2(a \mid b \mid c) = d$ in the promonoidal category implies $a_0 \fatsemi (b_0 \otimes c_0) \fatsemi a_1 = d_0$ in the monoidal category;
(4) $\varphi_0(a) = b$ in the promonoidal category implies $a_0 \fatsemi a_1 = b_0$ in the monoidal category.

On the other hand, a produoidal functor $\mathbb{B} \to \mathsf{mSplice}(\mathbb{M})$, also amounts to the following data. We will state it in multiple points and finally confirm that each one of these points has a correspondence on the first part of the proof, finishing the definition.

(1) For each object $X \in \mathbb{B}_{\mathrm{obj}}$, an object $(X^\circ, X^\bullet) \in \mathsf{mSplice}(\mathbb{M})_{\mathrm{obj}}$;
(2) for each element $f \colon \mathsf{N} \to X$, a morphism $f_0 \colon \mathsf{N} \to \binom{X^\circ}{X^\bullet}$;
(3) for each unit, $f \colon I \to X$, a unit $\langle f_0 \parallel f_1 \rangle \colon I \to \binom{X^\circ}{X^\bullet}$;
(4) for each morphism $f \colon X \to Y$, a splice $\langle f_0 \fatsemi \square \fatsemi f_1 \rangle \colon \binom{X^\circ}{X^\bullet} \to \binom{Y^\circ}{Y^\bullet}$;
(5) for each seq. join $f \colon X \triangleleft Y \to Z$, a spliced arrow

$$\langle f_0 \fatsemi \square \fatsemi f_1 \fatsemi \square \fatsemi f_2 \rangle \colon \binom{X^\circ}{X^\bullet} \triangleleft \binom{Y^\circ}{Y^\bullet} \to \binom{Z^\circ}{Z^\bullet};$$

(6) for each par. join $f \colon X \otimes Y \to Z$, a spliced arrow

$$\langle f_0 \fatsemi \square \otimes \square \fatsemi f_1 \rangle \colon \binom{X^\circ}{X^\bullet} \otimes \binom{Y^\circ}{Y^\bullet} \to \binom{Z^\circ}{Z^\bullet}.$$

The following conditions must hold for each one of the promonoidal categories. These correspond definitionally to the conditions for the two promonoidal structures we imposed before.

(1) $\alpha(a \mid b) = (c \mid d)$ in the produoidal category implies $\alpha(Fa \mid Fb) = (Fc \mid Fd)$ for the spliced arrows;
(2) $\lambda(a \mid b) = c = \rho(d \mid e)$ in the produoidal category implies $\lambda(Fa \mid Fb) = Fc = \rho(Fd \mid Fe)$ for the spliced arrows.

Finally, all the following conditions must also hold for the produoidal category. These are definitionally equal to the conditions for the produoidal category we imposed before.

(1) $\psi_2(a \mid b \mid c) = (d \mid e \mid f)$ in the produoidal category implies

$$\psi_2(Fa \mid Fb \mid Fc) = (Fd \mid Fe \mid Ff);$$

(2) $\psi_0(a) = (b \mid c \mid d)$ in the produoidal category implies

$$\psi_0(Fa) = (Fa \mid Fc \mid Fd);$$

(3) $\varphi_2(a \mid b \mid c) = d$ in the produoidal category implies

$$\varphi_2(Fa \mid Fb \mid Fc) = Fd;$$

(4) $\varphi_0(a) = b$ in the produoidal category implies $\varphi_0(Fa) = Fb$.

Each of these points is exactly equal by definition to the relative point in the first part of the proof: this establishes the desired adjunction. In other words, the definition of monoidal splice is precisely the one that makes this proof hold definitionally – even when we gave multiple different characterizations of it. □

3.3.4. A Representable Parallel Structure. A produoidal category has two tensors, and neither is, in principle, representable. However, the cofree produoidal category over a category we have just constructed happens also to have a representable tensor, $(\otimes)$: spliced monoidal arrows form a monoidal category.

Remark 3.3.11. This means $\mathsf{mSplice}(\mathbb{C})$ has the structure of a *virtual duoidal category* [Shu17] or *monoidal multicategory*, defined by Aguiar, Haim and López Franco [AHLF18] as a pseudomonoid in the cartesian monoidal 2-category of multicategories.

Proposition 3.3.12. *Parallel joins and parallel units of spliced monoidal arrows are representable profunctors. Explicitly,*

$$\mathsf{mSplice}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix} \otimes \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) \cong \mathsf{mSplice}(\mathbb{C})\left(\begin{smallmatrix} X \otimes X' \\ Y \otimes Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right), \text{ and}$$
$$\mathsf{mSplice}(\mathbb{C})\left(I; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) \cong \mathsf{mSplice}(\mathbb{C})\left(\begin{smallmatrix} I \\ I \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right).$$

In fact, these sets are equal by definition. However, we argue that there is a reason to work in the full generality of produoidal categories: produoidal categories can always be *normalized*.

Remark 3.3.13. Normalization is a procedure to mix both tensors of a duoidal category, $(\otimes)$ and $(\triangleleft)$, but not every duoidal category has a normalization [GF16]. It is folklore that one loses nothing by regarding non-representable produoidal structures as representable *duoidal structures on presheaves*, dismissing that they are moreover *closed* [Day70]; thus, one would expect only some produoidal categories to be normalizable – after all, only some duoidal categories are. Against folklore, we prove that every produoidal category, representable or not, has a *universal normalization*, a normal produoidal category which may be again representable or not.

3.3.5. Bibliography. Motivated by language theory and the representation theorem of Chomsky and Schützenberger, Melliès and Zeilberger [MZ22] were the first to present the multicategorical *splice-contour* adjunction. We are indebted to their exposition, which we extend to the promonoidal and produoidal cases. Our contribution is to show how monoidal contexts arise from an extended produoidal splice-contour adjunction; unifying these two threads.

Street already noted that the endoprofunctors of a monoidal category had a duoidal structure [Str12]; Pastro and Street described a promonoidal structure on lenses [PS07] and Garner and López-Franco contributed a partial normalization procedure for duoidal categories [GF16]. We build on top of this literature, putting it together, spelling out existence proofs, popularizing its produoidal counterpart and providing multiple new results and constructions that were previously missing (e.g. Theorems 3.3.10, 3.4.6 and 3.5.3).

This section takes its main ideas from joint work of this author with Matt Earnshaw and James Hefford [EHR23]. Earnshaw and Sobociński [ES22] have described a syntactic congruence on formal languages of string diagrams using monoidal contexts.

3.4. Interlude: Produoidal Normalization

3.4.1. Normal Produoidal Categories. Produoidal categories seem to contain too much structure: of course, we want to split things in two different ways, sequentially ($\lhd$) and in parallel ($\otimes$); but that does not necessarily mean that we want to keep track of two different types of units, parallel (I) and sequential (N). The atomic components of our decomposition algebra should be the same, without having to care if they are *atomic for sequential composition* or *atomic for parallel composition*.

Remark 3.4.1. The monoidal spliced arrows we just introduced are a perfect example: if we simply want a hole in a string diagram, the type it may take depends on the wires we want to leave to each side (see Figure 35). We would prefer to construct a new category – a Kleisli category on top of this one – where the bureaucracy of the units was already handled for us.

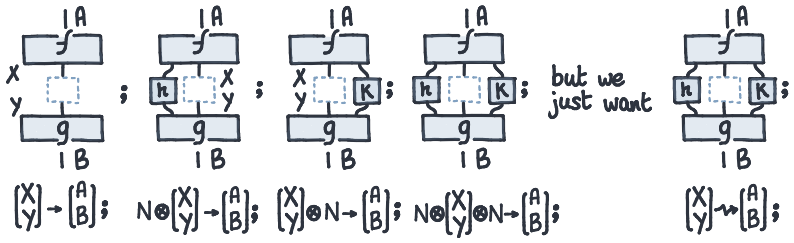


FIGURE 11. Multiple units complicate types.

3.4.2. The Normalization Monad. Fortunately, there exists an abstract procedure that, starting from any produoidal category, constructs a new produoidal category where both units are identified. This procedure is known as *normalization*, and the resulting produoidal categories are called *normal*.

Definition 3.4.2 (Normal produoidal category)**.** A *normal produoidal category* is a produoidal category where the unit interchanger $n\colon \mathbb{V}(I;\bullet) \to \mathbb{V}(\mathsf{N};\bullet)$ is an isomorphism.

Normal produoidal categories form a category with produoidal functors between them, **nProDuo**. As a consequence, it is endowed with a fully faithful forgetful functor $\mathcal{U}\colon$ **nProDuo** $\to$ **ProDuo**.

THEOREM 3.4.3. *Let $\mathbb{V}_{\otimes,I,\lhd,N}$ be a produoidal category. The profunctor*

$$\mathsf{Nor}(\mathbb{V})(\bullet;\bullet) = \mathbb{V}(\mathsf{N} \otimes \bullet \otimes \mathsf{N};\bullet)$$

forms a promonad. Moreover, the Kleisli category of this promonad is a normal produoidal category with the following profunctors.

(1) $\mathsf{Nor}(\mathbb{V})(X \otimes_{\mathsf{N}} Y; Z) = \mathbb{V}(\mathsf{N} \otimes X \otimes \mathsf{N} \otimes Y \otimes \mathsf{N}; Z)$*;*
(2) $\mathsf{Nor}(\mathbb{V})(X \lhd_{\mathsf{N}} Y; Z) = \mathbb{V}((\mathsf{N} \otimes X \otimes \mathsf{N}) \lhd (\mathsf{N} \otimes Y \otimes \mathsf{N}); Z)$*; and*
(3) $\mathsf{Nor}(\mathbb{V})(I_{\mathsf{N}}; X) = \mathsf{Nor}(\mathbb{V})(\mathsf{N}_{\mathsf{N}}; X) = \mathbb{V}(\mathsf{N}; X)$*.*

PROOF. Let us prove that $\mathsf{Nor}(\mathbb{V})$ is a promonad. We now define the multiplication and unit for the promonad, $\mathsf{Nor}(\mathbb{V})$. They are constructed out of laxators of the produoidal category $\mathbb{V}$ and Yoneda isomorphisms; thus, they must be associative and unital by coherence. The unit is defined by *(i)* unitality of $\mathbb{V}$, *(ii)* the laxator of $\mathbb{V}$, and *(iii)* by definition of $\mathsf{Nor}(\mathbb{V})$.

$$\mathbb{V}(X;Y) \cong \mathbb{V}(I \otimes X \otimes I; Y) \to \mathbb{V}(\mathsf{N} \otimes X \otimes \mathsf{N}; Y) \cong \mathsf{Nor}(\mathbb{V})(X;Y).$$

Multiplication is constructed as follows, using *(i)* the definition of $\mathsf{Nor}(\mathbb{V})$, *(ii)* Yoneda reduction, *(iii)* the laxators of $\mathbb{V}$, and *(iv)* the definition of $\mathsf{Nor}(\mathbb{V})$.

$$\begin{aligned}\textstyle\int^{Y\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(X;Y)\times\mathsf{Nor}(\mathbb{V})(Y;Z) &= \textstyle\int^{Y\in\mathbb{V}}\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N};Y)\times\mathbb{V}(\mathsf{N}\otimes Y\otimes\mathsf{N};Z)\\ &\cong \mathbb{V}(\mathsf{N}\otimes\mathsf{N}\otimes X\otimes\mathsf{N}\otimes\mathsf{N};Y)\to\mathbb{V}(\mathsf{N}\otimes Y\otimes\mathsf{N};Z)=\mathsf{Nor}(\mathbb{V})(X;Z).\end{aligned}$$

Associativity and unitality follow from those of the produoidal category.

The second part of this proof will show that $\mathsf{Nor}(\mathbb{V})$ is indeed a produoidal category: we will construct its associators, unitors and laxators from those of $\mathbb{V}$ and Yoneda isomorphisms. The right unitor is constructed as follows; the left unitor is constructed in a similar way: *(i)* by definition of $\mathsf{Nor}(\mathbb{V})$, *(ii)* by associativity, *(iii)* by definition, *(iv)* by Yoneda reduction, *(v)* by definition, and *(vi)* by unitality.

$$\begin{aligned}
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(\mathsf{N};M)\times\mathsf{Nor}(\mathbb{V})(X\otimes_{\mathsf{N}}M;Y) && =\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(\mathsf{N};M)\times\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N}\otimes M\otimes\mathsf{N};Y) && \cong\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V}),P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(\mathsf{N};M)\times\mathbb{V}(\mathsf{N}\otimes M\otimes\mathsf{N};P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;Y) && =\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V}),P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(\mathsf{N};M)\times\mathsf{Nor}(\mathbb{V})(M;P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;Y) && \cong\\
&\textstyle\int^{P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(\mathsf{N};P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;Y) && =\\
&\textstyle\int^{P\in\mathbb{V}}\mathbb{V}(\mathsf{N};P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;Y) && \cong\\
&\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N};Y).
\end{aligned}$$

Let us now construct the associator in two steps: we will show that both sides of the following equation

$$\begin{aligned}&\textstyle\int^{P\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;P)\times\mathsf{Nor}(\mathbb{V})(X\otimes_{\mathsf{N}}P;A) \;\cong\\ &\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(X\otimes_{\mathsf{N}}Y;M)\times\mathsf{Nor}(\mathbb{V})(M\otimes_{\mathsf{N}}Z;A)\end{aligned}$$

are isomorphic to $\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N}\otimes Y\otimes\mathsf{N}\otimes Z\otimes\mathsf{N};A)$. The first side is isomorphic by *(i)* definition of $\mathsf{Nor}(\mathbb{V})$, *(ii)* associativity of $\mathbb{V}$, *(iii)* definition of $\mathsf{Nor}(\mathbb{V})$, *(iv)* Yoneda reduction, *(v)* definition of $\mathsf{Nor}(\mathbb{V})$, and *(vi)* associativity. The second side is analogous.

$$\begin{aligned}
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;M)\times\mathsf{Nor}(\mathbb{V})(X\otimes_{\mathsf{N}}M;A) && =\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V})}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;M)\times\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N}\otimes M\otimes\mathsf{N};A) && \cong\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V}),P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;M)\times\mathbb{V}(\mathsf{N}\otimes M\otimes\mathsf{N};P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;A) && =\\
&\textstyle\int^{M\in\mathsf{Nor}(\mathbb{V}),P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;M)\times\mathsf{Nor}(\mathbb{V})(M;P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;A) && \cong\\
&\textstyle\int^{P\in\mathbb{V}}\mathsf{Nor}(\mathbb{V})(Y\otimes_{\mathsf{N}}Z;P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;A)\times && =\\
&\textstyle\int^{P\in\mathbb{V}}\mathbb{V}(\mathsf{N}\otimes Y\otimes\mathsf{N}\otimes Z\otimes\mathsf{N};P)\times\mathbb{V}(\mathsf{N}\otimes X\otimes P;A) && \cong\\
&\mathbb{V}(\mathsf{N}\otimes X\otimes\mathsf{N}\otimes Y\otimes\mathsf{N}\otimes Z\otimes\mathsf{N};A).
\end{aligned}$$

Finally, let us construct the four interchangers that define the produoidal category. Three of them are immediate: they are either identities or unitors: $\mathsf{Nor}(\mathbb{V})(I_{\mathsf{N}},A)\cong\mathsf{Nor}(\mathbb{V})(I_{\mathsf{N}}\triangleleft I_{\mathsf{N}};A)$ is the first, $\mathsf{Nor}(\mathbb{V})(\mathsf{N}\otimes\mathsf{N};A)\cong\mathsf{Nor}(\mathbb{V})(\mathsf{N};A)$ is the second, and $\mathsf{Nor}(\mathbb{V})(I_{\mathsf{N}};A)\cong\mathsf{Nor}(\mathbb{V})(\mathsf{N}_{\mathsf{N}};A)$ is the last one. We note here that this produoidal category is natural because of these isomorphisms. Thus, we only need to construct the first interchanger morphism of a produoidal category,

$$\mathsf{Nor}(\mathbb{V})((X_1\triangleleft Y_1)\otimes(X_2\triangleleft Y_2);A)\longrightarrow\mathsf{Nor}(\mathbb{V})((X_1\otimes X_2)\triangleleft(Y_1\otimes Y_2);A),$$

which is defined by the following reasoning. Here, we abbreviate $\mathsf{N}\otimes X\otimes\mathsf{N}$ by $X^{\otimes\mathsf{N}}$, and we apply *(i)* the definition of the tensors of $\mathsf{Nor}(\mathbb{V})$, *(ii)* the interchanger

of $\mathbb{V}$, *(iii)* unitality in $\mathbb{V}$, and *(iv)* the definition of tensors of $\mathsf{Nor}(\mathbb{V})$.

$$\begin{aligned}
&\mathsf{Nor}(\mathbb{V})((X_1 \lhd_{\mathsf{N}} Y_1) \otimes_{\mathsf{N}} (X_2 \lhd_{\mathsf{N}} Y_2); A) && = \\
&\mathbb{V}(\mathsf{N} \otimes (X_1^{\otimes \mathsf{N}} \lhd Y_1^{\otimes \mathsf{N}}) \otimes \mathsf{N} \otimes (X_2^{\otimes \mathsf{N}} \lhd Y_2^{\otimes \mathsf{N}}) \otimes \mathsf{N}; A) && \to \\
&\mathbb{V}((\mathsf{N} \otimes X_1^{\otimes \mathsf{N}} \otimes \mathsf{N} \otimes Y_1^{\otimes \mathsf{N}} \otimes \mathsf{N}) \lhd (\mathsf{N} \otimes X_2^{\otimes \mathsf{N}} \otimes \mathsf{N} \otimes Y_2^{\otimes \mathsf{N}} \otimes \mathsf{N}); A) && \to \\
&\mathbb{V}((\mathsf{N} \otimes X_1 \otimes \mathsf{N} \otimes Y_1 \otimes \mathsf{N})^{\otimes \mathsf{N}} \lhd (\mathsf{N} \otimes X_2 \otimes \mathsf{N} \otimes Y_2 \otimes \mathsf{N})^{\otimes \mathsf{N}}; A) && \to \\
&\mathsf{Nor}(\mathbb{V})((X_1 \otimes_{\mathsf{N}} Y_1) \lhd_{\mathsf{N}} (X_2 \otimes_{\mathsf{N}} Y_2); A). &&
\end{aligned}$$

The structure equations of the laxators follow from those of the base category $\mathbb{V}$. This finishes the construction of a produoidal category on the Kleisli category of the promonad. □

Lemma 3.4.4. *Normalization extends to an idempotent monad.*

Proof. The first part of the proof will show that $\mathsf{Nor}(\mathbb{V})$ is the free normal produoidal category over $\mathbb{V}$ by constructing the a monad structure on top of the functor $\mathsf{Nor}\colon \mathbf{nProDuo} \to \mathbf{nProDuo}$. Let us construct the unit and the multiplication of the monad. The unit $\eta_{\mathbb{V}} : \mathbb{V} \to \mathsf{Nor}(\mathbb{V})$ is defined as the identity-on-objects functor associated to the promonad; it and acts on morphisms by the unit of the promonad. Let us show that this is a produoidal functor by constructing the following components; all of them use the map $I \to \mathsf{N}$ from the base produoidal category,

(1) $\eta_{\otimes} : \mathbb{V}(X \otimes Y; A) \to \mathbb{V}(I \otimes X \otimes I \otimes Y \otimes I; A) \to \mathbb{V}(\mathsf{N} \otimes X \otimes \mathsf{N} \otimes Y \otimes \mathsf{N}; A)$;
(2) $\eta_I : \mathbb{V}(I; A) \to \mathbb{V}(\mathsf{N}; A)$;
(3) $\eta_{\lhd} : \mathbb{V}(X \lhd Y; A) \to \mathbb{V}((I \otimes X \otimes I) \lhd (I \otimes Y \otimes I); A) \to \mathbb{V}((\mathsf{N} \otimes X \otimes \mathsf{N}) \lhd (\mathsf{N} \otimes Y \otimes \mathsf{N}); A)$;
(4) $\eta_{\mathsf{N}} : \mathbb{V}(\mathsf{N}; A) \to \mathbb{V}(\mathsf{N}; A)$ is simply an identity.

these preserve laxators and coherence maps since they are constructed only from laxators and coherence maps.

Let us construct now the multiplication of the monad, $\mu_{\mathbb{V}} : \mathsf{Nor}(\mathsf{Nor}(\mathbb{V})) \to \mathsf{Nor}(\mathbb{V})$, and show that it is an isomorphism, making it an idempotent monad. The underlying functor is identity on objects, and it acts on morphisms by the normality of the already normalized produoidal category,

$$\mathsf{Nor}(\mathsf{Nor}(\mathbb{V}))(X; Y) = \mathsf{Nor}(\mathbb{V})(\mathsf{N} \otimes_{\mathsf{N}} X \otimes_{\mathsf{N}} \mathsf{N}; Y) \cong \mathsf{Nor}(\mathbb{V})(X; Y).$$

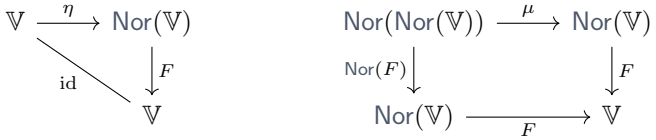
The following components make this functor a produoidal functor, they are constructed again from the normality of the already normalized produoidal category:

(1) $\mu_{\otimes} : \mathsf{Nor}(\mathsf{Nor}(\mathbb{V}))(X \otimes_{\mathsf{NN}} Y; A) = \mathsf{Nor}(\mathbb{V})(\mathsf{N} \otimes_{\mathsf{N}} X \otimes_{\mathsf{N}} \mathsf{N} \otimes_{\mathsf{N}} Y \otimes_{\mathsf{N}} \mathsf{N}; A) \cong \mathsf{Nor}(\mathbb{V})(X \otimes_N Y; A)$;
(2) $\mu_{\lhd} : \mathsf{Nor}(\mathsf{Nor}(\mathbb{V}))(X \lhd_{\mathsf{NN}} Y; A) = \mathsf{Nor}(\mathbb{V})((\mathsf{N} \otimes_{\mathsf{N}} X \otimes_{\mathsf{N}} \mathsf{N}) \lhd_{\mathsf{N}} (\mathsf{N} \otimes_{\mathsf{N}} Y \otimes_{\mathsf{N}} \mathsf{N}); A) \cong \mathsf{Nor}(\mathbb{V})(X \lhd_{\mathsf{N}} Y; A)$;
(3) $\mu_I : \mathsf{Nor}(\mathsf{Nor}(\mathbb{V}))(\mathsf{N}; A) = \mathsf{Nor}(\mathbb{V})(\mathsf{N}; A)$;
(4) $\mu_{\mathsf{N}} : \mathsf{Nor}(\mathsf{Nor}(\mathbb{V}))(\mathsf{N}; A) = \mathsf{Nor}(\mathbb{V})(\mathsf{N}; A)$,

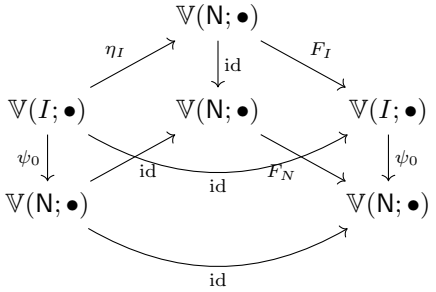
Finally we verify the monad laws. $\eta_{\mathcal{N}\mathbb{V}} \fatsemi \mu_{\mathbb{V}}$ is an identity-on-objects; on morphisms, it applies left and right unitors followed by their inverses; as a consequence, its underlying functor is the identity. The components of the natural transformations are also identities, since the interchanger $I \to \mathsf{N}$ is an identity for the normalized produoidal category $\mathsf{Nor}(\mathbb{V})$; they are otherwise composed of unitors followed by their inverses. The last step is to check the associativity of the monad, $\mu_{\mathsf{Nor}(\mathbb{V})} \fatsemi \mu_{\mathbb{V}}$ and $\mathsf{Nor}(\mu_{\mathbb{V}}) \fatsemi \mu_{\mathbb{V}}$; this is simply the identity on objects, so we simply apply left and right unitors twice on morphisms and their components. □

Lemma 3.4.5. *A produoidal category $\mathbb{V}$ has exactly one algebra structure for the normalization monad when it is normal, and none otherwise.*

Proof. Let $(F, F_\otimes, F_I, F_{\mathsf{N}}, F_{\mathsf{N}})\colon \mathsf{Nor}(\mathbb{V}) \to \mathbb{V}$ be an algebra. This means that the following commutative diagrams with the unit and multiplication of the normalization monad must commute.



Now, consider how the interchanger $\psi_0\colon \mathbb{V}(I;\bullet) \to \mathbb{V}(\mathsf{N};\bullet)$ is transported by these maps.



We conclude that $\eta_I = \psi_0$, but also that $F_{\mathsf{N}} = \mathrm{id}$. As a consequence, ψ_0 is invertible and F_I must be its inverse. We have shown that any produoidal category that is an algebra for the normalization monad must be normal.

We will now show that this already determines all of the functor F. We know that $\eta_\otimes, \eta_\triangleleft, \eta$ are isomorphisms because they are constructed from the unitors, associators, and the laxator ψ_0, which is an isomorphism in this case. This determines that $F_\otimes, F_\triangleleft, F$ must be their inverses. By construction, these satisfy all structure equations. □

Theorem 3.4.6 (Free normal produoidal). *Normalization determines an adjunction between produoidal categories and normal produoidal categories,*

$$\mathsf{Nor}\colon \mathbf{ProDuo} \rightleftharpoons \mathbf{nProDuo}\colon \mathsf{Forget}.$$

That is, $\mathsf{Nor}(\mathbb{V})$ *is the free normal produoidal category over* $\mathbb{V}$.

Proof. We know that the algebras for the normalization monad are exactly the normal produoidal categories (Lemma 3.4.5). We also know that the normalization monad is idempotent (Theorem 3.4.6). This implies that the forgetful functor from its category of algebras is fully faithful, and thus, the algebra morphisms are exactly the produoidal functors. As a consequence, the canonical adjunction to the category of algebras of the monad is exactly an adjunction to the category of normal produoidal categories. □

Remark 3.4.7. Garner and López Franco [GF16] introduced a partial normalization procedure for duoidal categories. We contribute a general normalization procedure for produoidal categories and we characterize it universally. Produoidal normalization behaves slightly better than duoidal normalization: it always succeeds, and we prove that it forms an idempotent monad (Theorem 3.4.6). The technical reason for this improvement is that the original duoidal normalization required the existence of certain coequalizers in $\mathbb{V}$; produoidal normalization uses coequalizers in **Set**.

In the previous Section 3.3.3, we constructed the produoidal category of spliced monoidal arrows, which distinguishes between morphisms and morphisms with a hole in the monoidal unit. This is because the latter hole splits the morphism in two parts. Normalization equates both; it sews these two parts. In Section 3.5, we explicitly construct monoidal contexts, the normalization of spliced monoidal arrows. Before that, let us also introduce the symmetric version of normalization.

3.4.3. Symmetric Normalization. Normalization is a generic procedure that applies to any produoidal category, it does not matter if the parallel join ($\otimes$) is symmetric or not. However, when $\otimes$ happens to be symmetric, we can also apply a more specialized normalization procedure: *symmetric normalization*.

Definition 3.4.8 (Symmetric produoidal category)**.** A *symmetric produoidal category* is a produoidal category $\mathbb{V}_{\lhd,N,\otimes,I}$ endowed with a natural isomorphism $\sigma\colon \mathbb{V}(X \otimes Y; Z) \cong \mathbb{V}(Y \otimes X; Z)$ satisfying the symmetry and hexagon equations.

THEOREM 3.4.9. *Let $\mathbb{V}$ be a symmetric produoidal category. The profunctor*

$$\mathsf{sNor}(\mathbb{V})(\bullet;\bullet) = \mathbb{V}(N \otimes \bullet; \bullet)$$

forms a promonad. Moreover, the Kleisli category of this promonad is a normal symmetric produoidal category with the following profunctors.

(1) $\mathsf{sNor}(\mathbb{V})(X \otimes_N Y; Z) = \mathbb{V}(N \otimes X \otimes Y; Z)$*;*
(2) $\mathsf{sNor}(\mathbb{V})(X \lhd_{\mathsf{N}} Y; Z) = \mathbb{V}((N \otimes X) \lhd (N \otimes Y); Z)$*; and*
(3) $\mathsf{sNor}(\mathbb{V})(I_{\mathsf{N}}; X) = \mathsf{sNor}(\mathbb{V})(\mathsf{N}_{\mathsf{N}}; X) = \mathbb{V}(\mathsf{N}; X)$.

THEOREM 3.4.10. *Normalization determines an adjunction between symmetric produoidal and normal symmetric produoidal categories,*

$$\mathsf{sNor}\colon \mathbf{symProDuo} \rightleftharpoons \mathbf{nSymProDuo}\colon \mathcal{U}.$$

That is, $\mathsf{sNor}(\mathbb{V})$ *is the free normal symmetric produoidal category over* $\mathbb{V}$.

3.4.4. Bibliography. Garner and López-Franco contributed a partial normalization procedure for duoidal categories [GF16], all of the credit for this elegant idea goes there.

We contribute its produoidal counterpart. The reader could think that this is an automatic process: extending an argument by Day [Day70], produoidal categories could be understood as closed duoidal categories in some sense. However, we show that there are some technical differences that make this case important: in the work of Garner and López-Franco, not every duoidal category has a normalization, and this prevents us from constructing a monad. We prove that produoidal normalization is always possible and defines an idempotent monad.

3.5. Monoidal Lenses

Monoidal lenses – the name we give to monoidal contexts [PGW17, Ril18, CEG+20] – formalize the notion of an incomplete morphism in a monoidal category. The category of monoidal lenses will have a rich algebraic structure: we shall be able to still compose contexts sequentially and in parallel and, at the same time, we shall be able to fill a context using another monoidal context. Perhaps surprisingly, then, the category of monoidal lenses is not even monoidal.

We justify this apparent contradiction in terms of profunctorial structure: the category is not monoidal, but it does have two promonoidal structures that precisely represent sequential and parallel composition. These structures form a normal produoidal category. In fact, we show it to be the normalization of the produoidal category of spliced monoidal arrows. This section constructs explicitly this normal produoidal category of monoidal lenses.

3.5.1. The Category of Monoidal Lenses. A monoidal lens – an element of type $\binom{X}{Y} \to \binom{A}{B}$ – represents a process from A to B with a hole admitting a process from X to Y. In this sense, monoidal lenses are similar to spliced monoidal arrows. The difference with spliced monoidal arrows is that monoidal lenses allow for communication to happen to the left and to the right of this hole.

Definition 3.5.1 (Monoidal lens)**.** Let $(\mathbb{C}, \otimes, I)$ be a monoidal category. *Monoidal lenses* are the elements of the profunctor

$$\mathsf{mLens}\,\binom{X}{Y}\binom{A}{B} = \int^{M_1, M_2} \mathbb{C}(A; M_1 \otimes X \otimes M_2) \times \mathbb{C}(M_1 \otimes Y \otimes M_2; B).$$

In other words, a *monoidal lens* from A to B, *with a hole* from X to Y, is an equivalence class consisting of a pair of objects $M, N \in \mathbb{C}_{\text{obj}}$ and a pair of morphisms $f \in \mathbb{C}(A; M \otimes X \otimes N)$ and $g \in \mathbb{C}(M \otimes Y \otimes N; B)$, quotiented by dinaturality of M and N.

Definition 3.5.2. Let $(\mathbb{C}, \otimes, I)$ be a monoidal category. Its normal produoidal category of *monoidal lenses*, $\mathsf{mLens}(\mathbb{C})$, has objects formed by pairs, $\mathsf{mLens}(\mathbb{C})_{obj} = (\mathbb{C}^{op} \times \mathbb{C})_{obj}$, and is defined by the following profunctors.

(1) Morphisms are diagrams with a single typed hole.

$$\mathsf{mLens}\mathbb{C}\,\binom{X}{Y}\binom{A}{B} = \int^{M_1, M_2} \mathbb{C}(A; M_1 \otimes X \otimes M_2) \times \mathbb{C}(M_1 \otimes Y \otimes M_2; B),$$

(2) Sequential joins are diagrams with a pair of sequential holes.

$$\mathsf{mLens}\mathbb{C}\left(\binom{X}{Y} \lhd \binom{X'}{Y'}; \binom{A}{B}; \right) = \int^{M_1, M_2} \mathbb{C}(A; M_1 \otimes X \otimes M_2) \times \mathbb{C}(M_1 \otimes Y \otimes M_2; M_3 \otimes X' \otimes M_4) \times \mathbb{C}(M_3 \otimes Y' \otimes M_4; B);$$

(3) Parallel joins are diagrams with a pair of parallel holes.

$$\mathsf{mLens}(\mathbb{C})\left(\binom{X}{Y} \otimes \binom{X'}{Y'}; \binom{A}{B}\right) = \int^{M_1, M_2, M_3} \mathbb{C}(A; M_1 \otimes X \otimes M_2 \otimes X' \otimes M_3) \times \mathbb{C}(M_1 \otimes Y \otimes M_2 \otimes Y' \otimes M_3; B)$$

(4) Units are complete diagrams with no holes, $\mathsf{mLens}(\mathbb{C})(\mathsf{N}; \binom{X}{Y}) = \mathbb{C}(X; Y)$.

Reading the profunctorial notation can be unenlightening. We provide the incomplete string diagrams for these profunctors in Figure 12 [Rom20b].

Theorem 3.5.3. *The category of monoidal lenses forms a normal produoidal category with its units, sequential and parallel joins. Monoidal lenses are the free*

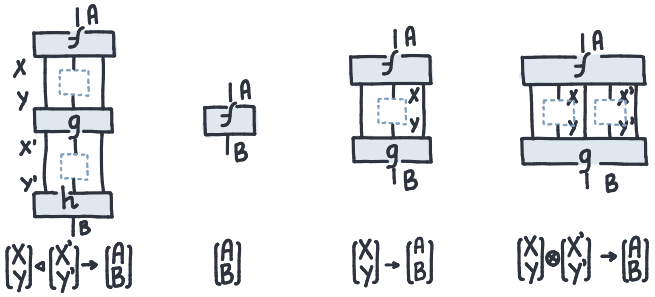


FIGURE 12. Monoidal lenses.

normalization of the cofree produoidal category over a category. In other words, monoidal lenses are the normalization of spliced monoidal arrows,

$$\mathsf{mLens}(\mathbb{C}) \cong \mathsf{Nor}(\mathsf{mSplice}(\mathbb{C})).$$

PROOF. The core of this result is in Theorem 3.4.6, which says that the normalization procedure yields the free normalization over a produoidal category. It is only left to check that this produoidal category of monoidal lenses that we have explicitly constructed in this section is precisely the normalization of the produoidal category of spliced arrows. We do so for morphisms, the rest of the proof is similar; the proof shows that $\mathsf{mSplice}(\mathbb{C})\left(\mathsf{N} \otimes \binom{X}{Y} \otimes \mathsf{N}; \binom{A}{B}\right)$, the normalization of spliced monoidal arrows, is isomorphic to monoidal lenses, $\mathsf{mLens}(\mathbb{C})\left(\binom{A}{B}; \binom{X}{Y}\right)$. We employ the Yoneda lemma on both V and V'.

$$\int^{U,V,U',V' \in \mathbb{C}} \mathbb{C}(A; U \otimes X \otimes U') \times \mathbb{C}(V \otimes Y \otimes V'; B) \times \mathbb{C}\left(U; V\right) \times \mathbb{C}\left(U'; V'\right) \quad \cong$$

$$\int^{U,U' \in \mathbb{C}} \mathbb{C}\left(A; U \otimes X \otimes U'\right) \times \mathbb{C}(U \otimes Y \otimes U'; B)$$

The rest of the profunctors follow a similar reasoning. □

3.5.2. Symmetric Monoidal Lenses. A symmetric monoidal lens of type $\mathsf{smLens}(\mathbb{C})\binom{X}{Y}\binom{A}{B}$ represents a process in a symmetric monoidal category with a hole admitting a process from X to Y.

Symmetric monoidal lenses are monoidal lenses, but we stop caring where the hole is. Again, the category of symmetric monoidal lenses has a rich algebraic structure; and again, most of this structure exists only virtually in terms of profunctors. In this case, though, the monoidal tensor *does* indeed exist: contrary to monoidal lenses, symmetric monoidal lenses form also a monoidal category. This is perhaps why applications of monoidal lenses have grown popular in recent years [Ril18], with applications in decision theory [GHWZ18], supervised learning [CGG+22, FJ19] and most notably in functional data accessing [Kme12, PGW17, BG18, CEG+20]. The promonoidal structure of optics was ignored, even when, after now identifying for the first time its relation to the monoidal structure of optics, we argue that it could be potentially useful in these applications: e.g. in multi-stage decision problems, or in multi-stage data accessors.

This section explicitly constructs the normal symmetric produoidal category of *symmetric monoidal lenses*. We describe it for the first time by a universal property: it is the free symmetric normalization of the cofree produoidal category.

Definition 3.5.4. Let $(\mathbb{C}, \otimes, I)$ be a symmetric monoidal category. *Symmetric monoidal lenses* are the elements of the profunctor

$$\mathsf{smLens}_{\mathbb{C}}\left(\begin{smallmatrix} X \\ Y \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \int^{M} \mathbb{C}(A; M \otimes X) \times \mathbb{C}(M \otimes Y; B).$$

In other words, a *symmetric monoidal lens* from A to B, *with a hole* from X to Y, is an equivalence class consisting of a pair of objects $M \in \mathbb{C}_{obj}$ and a pair of morphisms $f \in \mathbb{C}(A; M \otimes X)$ and $g \in \mathbb{C}(M \otimes Y; B)$, quotiented by dinaturality of M.

Definition 3.5.5. Let $(\mathbb{C}, \otimes, I)$ be a symmetric monoidal category. Its normal symmetric produoidal category of *symmetric monoidal lenses*, $\mathsf{smLens}(\mathbb{C})$, has objects formed by pairs, $\mathsf{smLens}(\mathbb{C})_{obj} = (\mathbb{C}^{op} \times \mathbb{C})_{obj}$, and is defined by the following profunctors.

(1) Morphisms are diagrams with a single typed hole.

$$\mathsf{smLens}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \int^{M} \mathbb{C}(A; M \otimes X) \times \mathbb{C}(M \otimes Y; B),$$

(2) Sequential joins are diagrams with a pair of sequential holes.

$$\mathsf{smLens}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix} \triangleleft \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \int^{M_1, M_2} \mathbb{C}(A; M_1 \otimes X) \times \mathbb{C}(M_1 \otimes Y; M_2 \otimes X') \times \mathbb{C}(M_2 \otimes Y'; B);$$

(3) Parallel joins are diagrams with a hole encompassing parallel wires.

$$\mathsf{smLens}(\mathbb{C})\left(\begin{smallmatrix} X \\ Y \end{smallmatrix} \otimes \begin{smallmatrix} X' \\ Y' \end{smallmatrix}; \begin{smallmatrix} A \\ B \end{smallmatrix}\right) = \int^{M} \mathbb{C}(A; M \otimes X \otimes X') \times \mathbb{C}(M \otimes Y \otimes Y'; B).$$

(4) Units are complete diagrams with no holes, $\mathsf{smLens}(\mathbb{C})(\mathsf{N}; \begin{smallmatrix} A \\ B \end{smallmatrix}) = \mathbb{C}(A; B)$.

Reading the profunctorial notation can be unenlightening. We provide the incomplete string diagrams for these profunctors in Figure 12 [Rom20b]. The only substantial difference with monoidal lenses is that we do not need to keep track of where the hole is placed.

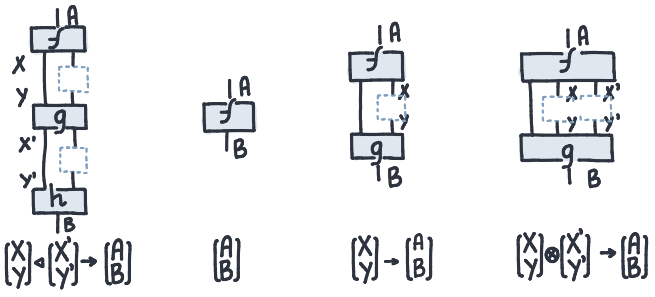


FIGURE 13. Symmetric monoidal lenses.

The term "monoidal lenses" has usually been reserved for the morphisms of this category; in the literature, the sequential splits – the lenses with multiple holes – get a name from their distinctive shape: these are *combs* or *quantum combs* [CFS16].

3.5.3. Towards Message Theories. Lenses, or *combs*, can be interpreted as incomplete morphisms, but also as morphisms that send and receive resources. The next chapter will exploit this intuition.

Remark 3.5.6 (Session notation for combs)**.** We will write $A^\circ = \binom{A}{I}$ and $B^\bullet = \binom{I}{B}$ for the objects of the symmetric produoidal category of lenses that have a monoidal unit as one of its objects. Thanks to $A^\circ \otimes B^\bullet = \binom{A}{B}$, these are enough to express all objects.

Proposition 3.5.7. *Let $(\mathbb{C}, \otimes, I)$ be a symmetric monoidal category. There exist monoidal functors*

$$(^\circ)\colon \mathbb{C} \to \mathsf{smLens}(\mathbb{C}), \quad \textit{and} \quad (^\bullet)\colon \mathbb{C}^{op} \to \mathsf{smLens}(\mathbb{C}).$$

Moreover, they satisfy the following properties definitionally: $\mathbb{C}(A^\bullet \triangleleft B^\bullet; \bullet) \cong \mathbb{C}(A^\bullet \otimes B^\bullet; \bullet)$; $(A \otimes B)^\circ = A^\circ \otimes B^\circ$; $\mathbb{C}(A^\circ \triangleleft B^\circ; \bullet) \cong \mathbb{C}(A^\circ \otimes B^\circ; \bullet)$; $(A \otimes B)^\bullet = A^\bullet \otimes B^\bullet$; and $\mathbb{C}(A^\circ \triangleleft B^\bullet; \bullet) \cong \mathbb{C}(A^\circ \otimes B^\bullet; \bullet)$.

Proof. We define $f^\circ = (f \fatsemi \square \fatsemi \mathrm{id}_I)$ and $g^\bullet = (\mathrm{id}_I \fatsemi \square \fatsemi g)$, and then check that compositions and tensoring of morphisms are compatible with composition and tensoring of monoidal lenses, this is straightforward. Moreover, we can see that, by definition,

$$(A \otimes B)^\circ = \tbinom{A \otimes B}{I} = \binom{A}{I} \otimes \binom{B}{I} = A^\circ \otimes B^\circ, \quad \text{and}$$
$$(A \otimes B)^\bullet = \tbinom{I}{A \otimes B} = \binom{I}{A} \otimes \binom{I}{B} = A^\bullet \otimes B^\bullet.$$

This proof appears with a different language in the work of Riley [Ril18, Proposition 2.0.14]. In fact, there, the combined identity-on-objects functor $(\circ \times \bullet)\colon \mathbb{C} \times \mathbb{C}^{op} \to \mathsf{smLens}(\mathbb{C})$ is shown to be monoidal. □

Example 3.5.8. Let us give a first example of how to employ combs for the description of concurrent protocols. Broadbent and Karvonen [BK22] propose a formalization of the *one-time pad* encryption protocol in a symmetric monoidal category endowed with a Hopf algebra with an integral.

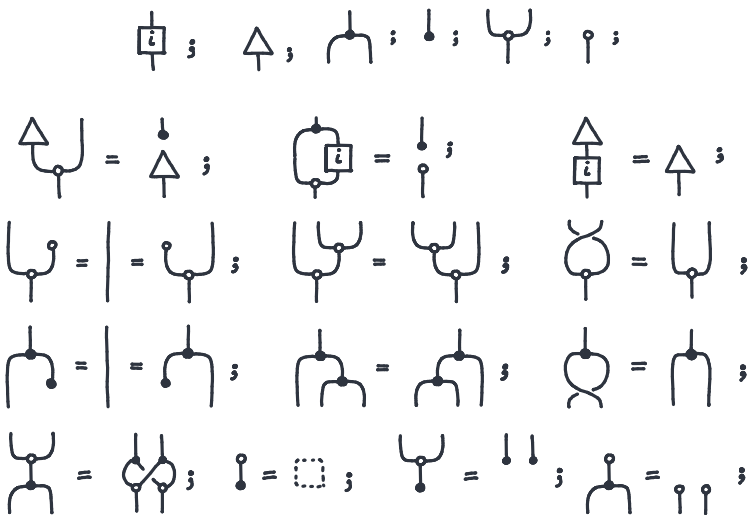


Figure 14. Theory of a Hopf algebra with an integral.

Definition 3.5.9. A Hopf algebra with an integral, $(X, \curlywedge, \curlyvee, \Y, \curlyvee, i, d)$, is a commutative bialgebra endowed with an *antipode* map $i\colon X \to X$, representing inversion; and endowed with an *integral map*, $d\colon I \to X$, representing a non-determined value, or pure noise. These must satisfy the equations in Figure 14.

The one-time pad is a mathematically secure encryption technique. It works as follows: *(i)* the two parties communicating – say, Alice and Bob – start by preparing some random bits and sharing them; *(ii)* when the message is ready, Alice applies bitwise XOR with the random bits to encrypt the message, and then broadcasts the encrypted message – a potential attacker, Eve, will receive this encrypted message; *(iii)* finally, Bob receives the encrypted message and applies again bitwise XOR with the random bits to decrypt the message (Figure 15).

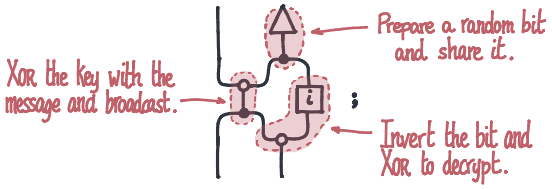


FIGURE 15. Description of the one-time pad.

Proposition 3.5.10. *The one-time pad is* secure*, meaning that it is equal to the process that sends a message from* Alice *to* Bob *and outputs random noise through the attacker's channel.*

PROOF. We repeat the proof from Broadbent and Karvonen [BK22]. We employ string diagrams of symmetric monoidal categories, in Figure 16, to show that the morphism is equal to an identity tensored by the integral of the Hopf algebra. □

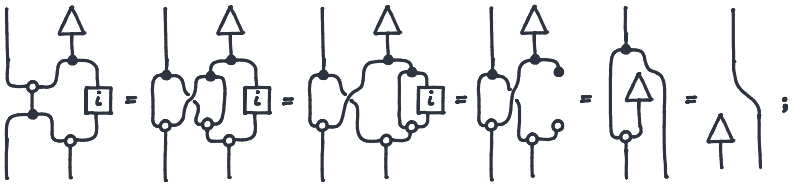


FIGURE 16. Correctness of the one-time pad.

The interesting part comes when we want to split the morphism into its different constituents: there should be a *stage* where the three actors play; Alice does not control the fact that the encrypted message will be broadcast; Eve, the attacker, can only attack at the end; Bob will need to keep a bit in memory. These considerations are part of the problem statement the one-time pad is solving. It is easy to come up with a morphism that connects an input to an output: the problem the one-time pad is solving is to do so on a stage that has been preset.

The components of the one time pad are not simply morphisms of a monoidal category (Figure 17). They must be understood as monoidal lenses. After this section, we can declare that a possible typing for these components is the following:

(1) $\mathsf{Alice}\colon \mathsf{N} \to \binom{X \otimes X}{X}$;

(2) Bob: $X^\bullet \triangleleft X^\circ \to \binom{I}{X}$;
(3) Eve: $\mathsf{N} \to \binom{X}{X}$;
(4) Stage: $X^\circ \triangleleft X^\bullet \triangleleft X^\bullet \triangleleft X^\circ \triangleleft X^\bullet \triangleleft X^\bullet \triangleleft X^\circ \triangleleft X^\circ \to \binom{X}{X \otimes X}$;

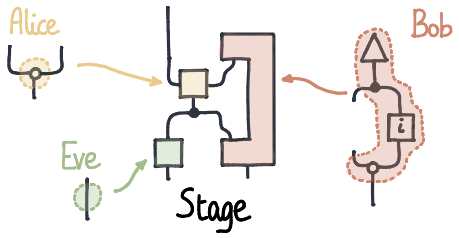


FIGURE 17. Components of the one-time pad.

Still, at this stage it is not easy to talk about message passing with this syntax. It is true that the new produoidal types can track all the exchanges that happen along a boundary; but these types are tedious – see, for instance, the long type of Stage – and it is not clear how to compose them. What we are missing is a combinatorial description of the different ways we can combine elements of this produoidal algebra.

This is what the next chapter will solve: we will propose a combinatorial algebra of message passing and show that lenses, the normalized cofree produoidal algebra over a symmetric monoidal category have a second universal property – they also constitute the free message theory.

3.5.4. Bibliography. Lenses [FGM+07] are a notion of bidirectional transformation that can be cast in arbitrary monoidal categories. The first mention of monoidal lenses separate from their classical database counterparts [JRW12] is due to Pastro and Street [PS07], who identify them as an example of a promonoidal category. However, it was with a different monoidal structure [Ril18] that they became popular in recent years, spawning applications not only in bidirectional transformations [FGM+07] but also in functional programming [PGW17, CEG+20], open games [GHWZ18], polynomial functors [NS22] and quantum combs [HC22]. Relating this monoidal category of lenses with the previous promonoidal category of lenses was an open problem; and the promonoidal structure was mostly ignored in applications. We solve this problem, proving that lenses are a universal normal symmetric produoidal category (the symmetric monoidal lenses), which endows them with a novel algebra and a novel universal property. This also extends work on the relation between *incomplete diagrams*, *comb-shaped diagrams*, and *lenses* [Rom20a, Rom20b].

Lenses themselves have been applied to protocol specification [VC22]. Spivak [Spi13] also discusses the multicategory of *wiring diagrams*, later used for incomplete diagrams [PSV21] and related to lenses [SSV20]; we conjecture that this multicategory of wiring diagrams is precisely the produoidal category of lenses, once we stop tracking dependencies explicitly.

Conjecture 3.5.11. *Each physical produoidal category induces a multicategory given by its physical lax tensor (Section 3.2.3). The multicategory of wiring diagrams [Spi13] of symmetric monoidal categories is the multicategory induced by the produoidal category of monoidal lenses.*

The promonoidal categories we use can be seen as multicategories with an extra coherence property. In this sense, we contribute the missing algebraic structure of the universal multicategory of *wiring diagrams relative to a monoidal category*.

CHAPTER 4

Monoidal Message Passing

Monoidal Message Passing

This chapter develops message passing in monoidal categories following the theory of context we just constructed. We have already defined what incomplete morphisms in monoidal categories are: we will now study the structure of all their possible compositions. This includes not only the obvious operations of composition but any possible *wiring* of a diagram that could combine them while respecting the acyclicity of string diagrams.

Studying seriously the combinatorial structure of string diagram composition arrives at the same conclusion as axiomatizing a naive theory of message passing: message theories. Indeed, we prove that the polarized shufflings that describe string diagram composition have as algebras precisely the message theories.

Section 4.1 introduces our minimalistic theory of message passing, with axioms that should hopefully be acceptable to any reader. Section 4.2 starts developing the categorical semantics for message theories, based on physical monoidal multicategories (a variant of duoidal categories); it then shows that shufflings from the free physical monoidal multicategory. Section 4.3 provides the second ingredient for this categorical semantics: polarization. We then combine both ingredients in Section 4.4: polar shuffles form the combinatorial structure that combines incomplete string diagrams; message theories are precisely the algebras of physical monoidal multicategory of polar shuffles. This chapter ends with an adjunction between message theories and symmetric monoidal categories, which ensures that we can construct a free message theory on top of any symmetric monoidal category.

4.1. Message Theories

4.1.1. Message Theories. Message passing requires the interplay of at least two mathematical structures: the ability to *interleave* events in time and the ability to connect a *sender and a receiver*. Let us propose a minimally axiomatized algebra of interleaving and sending/receiving: interleaving will correspond to a normal duoidal algebra, and sending/receiving will correspond to polarization.

Definition 4.1.1. A *message theory* $\mathbb{M}$ consists of a set of types, $\mathbb{M}_{obj}$ with extra structure: a *send/receive session type* is a polarized list of types; for each session type, we have a collection of *sessions* with that type,

$$\mathbb{M}(X_1^{\bullet_1}, \dots, X_n^{\bullet_n}), \text{ for each } X_1, \dots, X_n \in \mathbb{M}_{obj}, \text{ and each polarization } \bullet_i \in \{\circ, \bullet\}.$$

A message theory must contain operations for *(i)* binary shuffling, *(ii)* and nullary shuffling, *(iii)* linking a sent message to immediately receive it, and *(iv)* spawning a channel that receives a message and sends it immediately.

(1) $\textsc{shf}_\sigma \colon \mathbb{M}(\Gamma) \times \mathbb{M}(\Delta) \to \mathbb{M}(\sigma(\Gamma, \Delta))$, shuffling two processes;
(2) $\textsc{nop} \colon \mathbb{M}()$, a no-operation, doing nothing;
(3) $\textsc{lnk}_x^{\Gamma;\Delta} \colon \mathbb{M}(\Gamma, X^\bullet, X^\circ, \Delta) \to \mathbb{M}(\Gamma, \Delta)$, linking send to receive;
(4) $\textsc{spw}_x^{\Gamma;\Delta} \colon \mathbb{M}(\Gamma, \Delta) \to \mathbb{M}(\Gamma, X^\circ, X^\bullet, \Delta)$, a receive to send channel.

Message theories may be better understood in the notation of a logic, as in Figure 1. Types form a free polarized monoid; each term describes a possible communication protocol.

$$\frac{\Gamma \qquad \Delta}{[\Gamma, \Delta]_\sigma}\ (\textsc{shf}_\sigma) \qquad \frac{\Gamma, X^\bullet, X^\circ, \Delta}{\Gamma, \Delta}\ (\textsc{lnk}) \qquad \frac{\Gamma, \Delta}{\Gamma, X^\circ, X^\bullet, \Delta}\ (\textsc{spw}) \qquad \frac{}{\varepsilon}\ (\textsc{nop})$$

FIGURE 1. Type-theoretic presentation of a message theory.

A message theory must satisfy the following axioms: *(i)* shuffles compose as in their symmetric malleable multicategory, where we write $(\sigma \fatsemi_1 \tau)$ to be the same by associativity as $(\tau' \fatsemi_2 \sigma')$, we write $(*)$ for the trivial shuffle, and we write $\tilde{\sigma}$ for the symmetric counterpart of σ; *(ii)* linking is natural with respect to the shuffles; *(iii)* spawning is natural with respect to the shuffles; *(iv)* linking is dual to spawning; and *(v)* independent linkings and spawnings commute.

(1a) $\textsc{shf}_\tau(\textsc{shf}_\sigma(m_1 m_2), m_3) = \textsc{shf}_{\sigma'}(m_1, \textsc{shf}_{\tau'}(m_2, m_3))$;
(1b) $\textsc{shf}_*(m, \textsc{nop}) = m$;
(1c) $\textsc{shf}_\sigma(m_1, m_2) = \textsc{shf}_{\tilde{\sigma}}(m_2, m_1)$;
(2a) $\textsc{shf}_{\sigma,\tau}(\textsc{lnk}_x^{\Gamma_1,\Gamma_2}(m_1), m_2) = \textsc{lnk}_x^{\Gamma_1,\Delta_1;\Gamma_2,\Delta_2}(\textsc{shf}_{\sigma,x,\tau}(m_1, m_2))$;
(2b) $\textsc{shf}_{\sigma,\tau}(m_1, \textsc{lnk}_x^{\Delta_1,\Delta_2}(m_2)) = \textsc{lnk}_x^{\Gamma_1,\Delta_1;\Gamma_2,\Delta_2}(\textsc{shf}_{\sigma,x,\tau}(m_1, m_2))$;
(3a) $\textsc{shf}_{\sigma,\tau}(\textsc{spw}_x^{\Gamma_1,\Gamma_2}(m_1), m_2) = \textsc{spw}_x^{\Gamma_1,\Delta_1;\Gamma_2,\Delta_2}(\textsc{shf}_{\sigma,\tau}(m_1, m_2))$;
(3b) $\textsc{shf}_{\sigma,\tau}(m_1, \textsc{spw}_x^{\Delta_1,\Delta_2}(m_2)) = \textsc{spw}_x^{\Gamma_1,\Delta_1;\Gamma_2,\Delta_2}(\textsc{shf}_{\sigma,\tau}(m_1, m_2))$;
(4a) $\textsc{lnk}_x^{\Gamma,X^\circ;\Delta}(\textsc{spw}_x^{\Gamma;X^\circ,\Delta}(m)) = m$;
(4b) $\textsc{lnk}_x^{\Gamma;X^\bullet,\Delta}(\textsc{spw}_x^{\Gamma,X^\bullet;\Delta}(m)) = m$;
(5a) $\textsc{lnk}_x^{\Gamma_1;\Gamma_2 Y \Gamma_3}(\textsc{spw}_y^{\Gamma_1 X \Gamma_2;\Gamma_3}(m)) = \textsc{spw}_y^{\Gamma_1\Gamma_2;\Gamma_3}(\textsc{lnk}_x^{\Gamma_1;\Gamma_2\Gamma_3}(m))$;
(5b) $\textsc{lnk}_y^{\Gamma_1 X \Gamma_2;\Gamma_3}(\textsc{spw}_x^{\Gamma_1;\Gamma_2 Y \Gamma_3}(m)) = \textsc{spw}_x^{\Gamma_1;\Gamma_2\Gamma_3}(\textsc{lnk}_y^{\Gamma_1 X \Gamma_2;\Gamma_3}(m))$;
(5c) $\textsc{spw}_x^{\Gamma_1;\Gamma_2 Y \Gamma_3}(\textsc{spw}_y^{\Gamma_1\Gamma_2;\Gamma_3}(m)) = \textsc{spw}_y^{\Gamma_1 X \Gamma_2;\Gamma_3}(\textsc{spw}_x^{\Gamma_1;\Gamma_2\Gamma_3}(m))$;
(5d) $\textsc{lnk}_x^{\Gamma_1;\Gamma_2\Gamma_3}(\textsc{lnk}_y^{\Gamma_1 X \Gamma_2;\Gamma_3}(m)) = \textsc{lnk}_y^{\Gamma_1\Gamma_2;\Gamma_3}(\textsc{lnk}_x^{\Gamma_1;\Gamma_2 Y \Gamma_3}(m))$.

These axioms are again better understood in logic notation, as equations between derivations, see Figure 2.

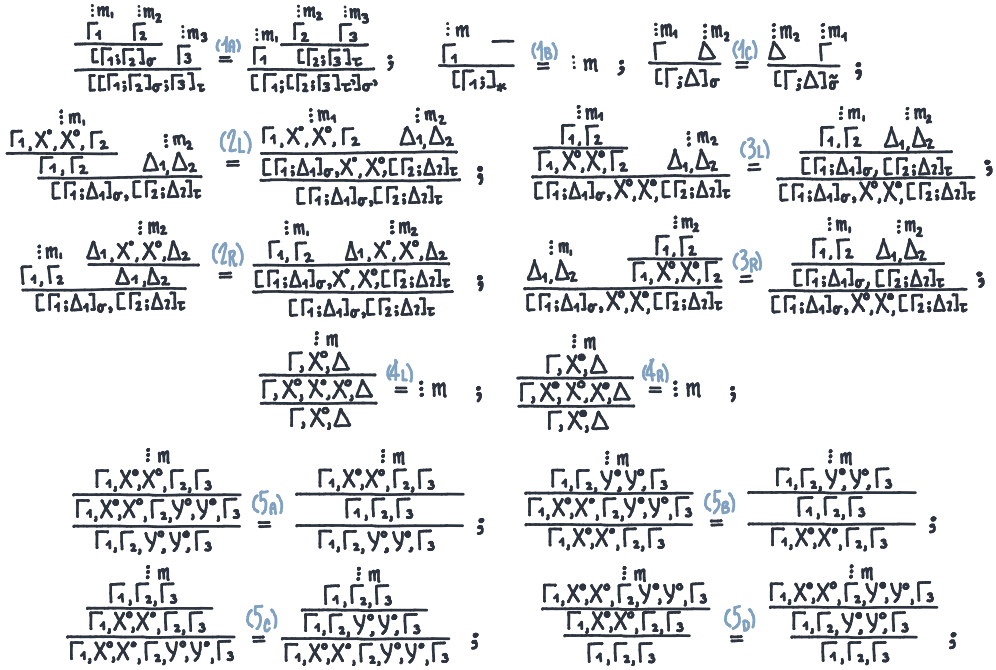


FIGURE 2. Axioms of a message theory.

Definition 4.1.2. A *message functor* $F\colon \mathbb{M} \to \mathbb{N}$ between two message theories, $\mathbb{M}$ and $\mathbb{N}$, is a function on objects $F_{obj}\colon \mathbb{M}_{obj} \to \mathbb{N}_{obj}$ that extends to a family of functions on session sets,

$$F\colon \mathbb{M}(X_1^\bullet, \dots, X_n^\bullet) \to \mathbb{N}(FX_1^\bullet, \dots, FX_n^\bullet).$$

This function must *(i)* preserve shuffling, $F(\textsc{shf}_\sigma(f,g)) = \textsc{shf}_\sigma(Ff, Fg)$; *(ii)* preserve spawning, $F(\textsc{spw}_x) = \textsc{spw}_{Fx}$; *(iii)* connecting, $F(\textsc{lnk}_x(f)) = \textsc{lnk}_x(Ff)$; and *(iv)* the no-operation, $F(\textsc{nop}) = \textsc{nop}$. Message theories form a category **Msg** with message functors between them.

4.1.2. Properties of a Message Theory. Our goal is to prove a coherence theorem for message theories. Building up to this result, let us reason with message theories to understand the basic properties that can be derived from the axioms.

Proposition 4.1.3. *Message theories have a derived operation for each shuffling; moreover, these operations compose as in the multicategory of shufflings.*

$$\mathsf{shuf}_\sigma^{\Gamma_1,\dots,\Gamma_n}\colon \mathbb{M}(\Gamma_1) \times \dots \times \mathbb{M}(\Gamma_n) \to \mathbb{M}([\Gamma_1, \dots, \Gamma_n]_\sigma)$$

PROOF. We have defined an operation for binary and nullary shufflings, and we have defined them to compose exactly as shufflings do. Because shufflings form a malleable multicategory, each n-ary shuffling can be recovered uniquely from the binary and nullary shufflings. □

Proposition 4.1.4. *A session can always send later and receive sooner, but it cannot send sooner nor receive later. Formally, there exist derived operations*

$$\mathsf{wait}_X^{\Gamma,\Delta,\Psi}\colon \mathbb{M}(\Gamma, X^\bullet, \Delta, \Psi) \to \mathbb{M}(\Gamma, \Delta, X^\bullet, \Psi),$$
$$\mathsf{rush}_X^{\Gamma,\Delta,\Psi}\colon \mathbb{M}(\Gamma, \Delta, X^\circ, \Psi) \to \mathbb{M}(\Gamma, X^\circ, \Delta, \Psi).$$

$$\dfrac{\dfrac{\Gamma, X^\bullet, \Delta, \Psi \qquad \overline{X^\circ, X^\bullet}\ (\text{SPW})}{\Gamma, X^\bullet, X^\circ, \Delta, X^\bullet, \Psi}\ (\text{SHF})}{\Gamma, \Delta, X^\bullet, \Psi}\ (\text{COM}) \qquad \dfrac{\dfrac{\Gamma, \Delta, X^\circ, \Psi \qquad \overline{X^\circ, X^\bullet}\ (\text{SPW})}{\Gamma, X^\circ, \Delta, X^\circ, X^\bullet, \Psi}\ (\text{SHF})}{\Gamma, X^\circ, \Delta, \Psi}\ (\text{COM})$$

FIGURE 3. Derivation of wait and rush.

PROOF. We can construct the derivation trees of both operations. They both spawn a new channel, shuffle its ends to the origin and target position, and they connect the channel.

The explicit construction is in Figure 3. Note how, thanks to polarization, it is not possible to use the same technique to *send sooner* nor *receive later*. In fact, we can reason by contradiction that sending sooner, $\mathbb{M}(\Gamma, \Delta, X^\bullet, \Psi) \to \mathbb{M}(\Gamma, X^\bullet, \Delta, \Psi)$, is impossible: the only possible operations we can apply in an arbitrary message theory to an arbitrary session are *shufflings with a spawned channel* and *connections*; we require at least a connection to eliminate the $X^\bullet$, but because the X° must come from shuffling a spawned channel, the corresponding $X^\bullet$ must be placed strictly after it. □

Proposition 4.1.5. *In particular, we can always swap the order of objects with the same polarity,*

$$\mathsf{swap}_\circ^{\Gamma;X;Y;\Delta} \colon \mathbb{M}(\Gamma, X^\circ, Y^\circ, \Delta) \to \mathbb{M}(\Gamma, Y^\circ, X^\circ, \Delta),$$
$$\mathsf{swap}_\bullet^{\Gamma;X;Y;\Delta} \colon \mathbb{M}(\Gamma, X^\bullet, Y^\bullet, \Delta) \to \mathbb{M}(\Gamma, Y^\bullet, X^\bullet, \Delta).$$

These are self-inverses, forming not only braidings but symmetries in the underlying monoidal category of positively or negatively polarized objects.

PROOF. Swaps are constructed from rushing and waiting (Proposition 4.1.4); explicitly,

$$\mathsf{swap}_\circ^{\Gamma;X;Y;\Delta} = \mathsf{rush}_Y^{\Gamma,X^\circ,\Psi}, \quad \text{and} \quad \mathsf{swap}_\bullet^{\Gamma;X;Y;\Delta} = \mathsf{wait}_Y^{\Gamma,Y^\bullet,\Psi}.$$

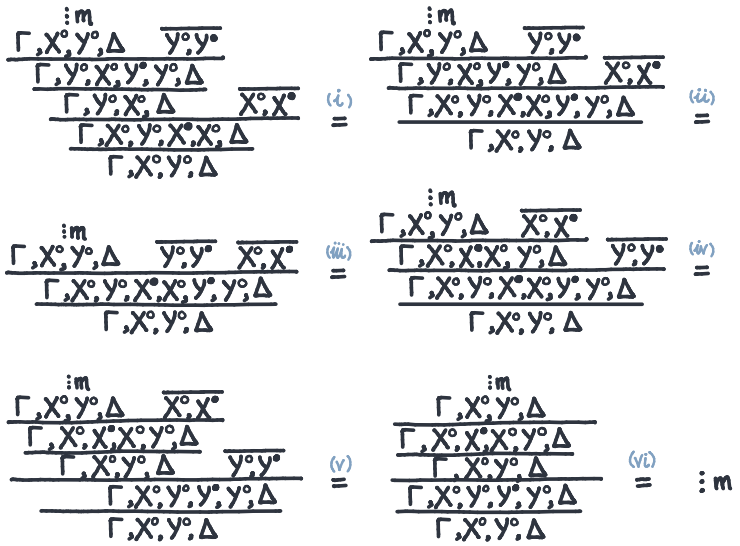


FIGURE 4. Swaps are self-inverses.

Let us prove that they are self-inverses; we do so with the negatively-polarized one (Figure 4), the other case is analogous. We reason using naturality of linking, *(i,iv)*, that shuffles compose as shuffles *(ii,iii)* (Proposition 4.1.3), the interaction between shuffling and spawning *(v)*, and the duality between spawning and linking *(vi)*. □

Proposition 4.1.6. *We can link sooner or later without changing the result. Formally, the equations in Figure 5 hold.*

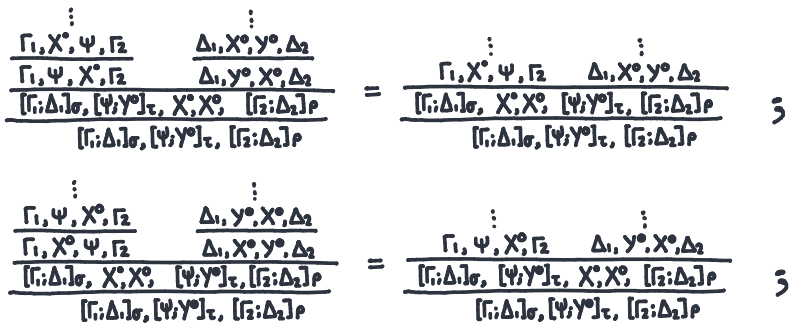


FIGURE 5. Linking sooner or later does not change the restult.

PROOF. We will prove the first one (Figure 6), the second one follows analogously. We reason using *(i)* the definition of wait; *(ii,iii)* that shuffles compose as shuffles (Proposition 4.1.3); *(iv)* naturality of linking; and *(v)* that swaps are self-inverses (Proposition 4.1.5). □

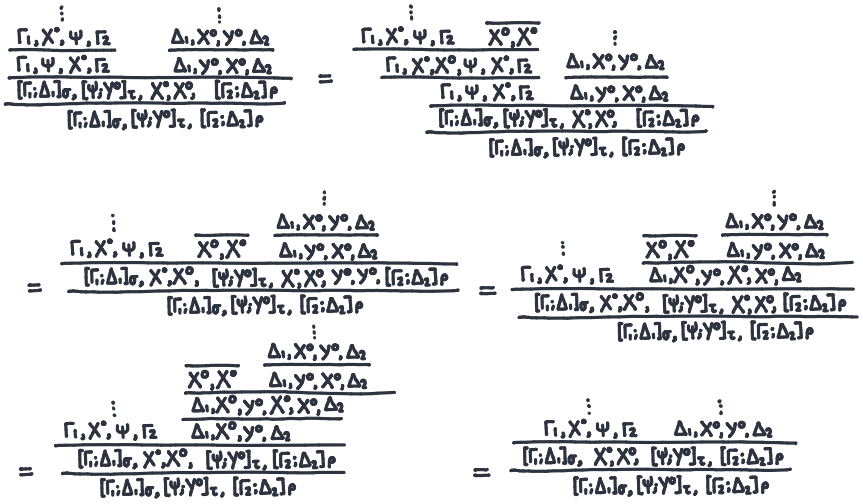


FIGURE 6. Proof of Proposition 4.1.6.

Proposition 4.1.7. *Any spawning factors as the spawning of a single channel followed by a shuffling. Formally,*

$$\mathrm{SPW}_X^{\Gamma,\Delta}(m) = \mathrm{SHF}_{\Gamma,X^\circ,X^\bullet,\Delta}(m, \mathrm{SPW}^i_X(\mathrm{NOP})).$$

PROOF. This is a direct consequence of Axioms (3a,3b). □

Theoretically, it would be possible to reason with message theories at the level of derivations. However, as we have done through this text, we will try to find better categorical semantics and a better combinatorial expression of the free message theory. We will be most interested in the categorical semantics of message theories and how do they interplay with process theories, in the sense of monoidal categories. We introduce specialized semantics in terms of physical monoidal multicategories, which are another instance of the idea of partially representing a physical duoidal category. For all of this, we will need a coherence theorem.

4.1.3. Coherence for Message Theories. Symmetric monoidal categories are not *perfectly coherent*: easily, we can find that there are two formally well-typed structure maps $A \otimes A \to A \otimes A$, the identity and the swap. However, what is indeed true is that any two *distinctly typed* structure maps in the free symmetric monoidal category are equal. "Distinct typing" is a notion that only makes sense in the free symmetric monoidal category over some generators; it means that the generators comprising the lists that are our objects appear only once with each variance: arrows $A \otimes B \otimes C \to B \otimes C \otimes A$ are distinctly typed, but arrows $A \otimes A \to A \otimes A$ are not, because A appears twice with each variance.

Shufflings satisfy a similar form of coherence: there is a unique way of shuffling two words into a third one if these words are distinctly typed. This section proves that message theories satisfy the same form of coherence. It is not true that any two parallel formal arrows are equal in any message theory: for instance, there are two ways of deriving X°, X° from X° and X°. It is true, however, that there is a unique arrow for any distinctly typed domains and codomain.

THEOREM 4.1.8. *Message theories are coherent. In the free message theory over a set of objects, there is at most a single derivation between any distinctly typed premises and conclusion.*

PROOF. Consider distinctly typed premises and conclusion. There must be three different classes of types in this derivation: *(1)* those that appear twice on the conclusions with different polarity, *(2)* those that appear twice on the premises with different polarity, and *(3)* those that appear once in the premises and once in the conclusions with the same polarity.

We will construct a non-unique normal form for derivations in a message theory, taking into account each one of these cases.

(1) In the first case, the types must have been created by spawning a channel. Using naturality of spawning (Axioms 3a, 3b, 5c), we can move these spawning operations to be shuffled at the end of the derivation. Note that it is not true that we can move *all* spawning to the end of the derivation, but if the types appear only in the conclusions, then we can always do so.
(2) In the second case, these variables must get linked to each other. We can always move the linkings to the end of the derivation (before spawning new objects) using again the naturality of the linkings (Axioms 2a, 2b, 5a).
(3) For the third case, only a shuffling, rushing and waiting can be involved (any linking or spawning that does not involve rushing or waiting has already been moved to the end). Rushing and waiting can be moved to the beginning of the derivation because of naturality of spawning and linking (Axioms 2a, 2b, 3a, 3b).

All of this argues that we can always factor a derivation in a message theory as *(i)* rushing and waiting, *(ii)* a shuffle, *(iii)* linkings, *(iv)* spawnings and shufflings of new variables; but we have not yet shown that this derivation is unique.

Imagine that we know the premises $(\Gamma_1, \dots, \Gamma_n)$ and the conclusion (Δ) of a derivation in a message theory. Let us argue that there is at most a unique derivation between these two.

(1) Removing the objects that appear twice in the conclusion (Δ), we obtain a new conclusion (Δ_1); there is, at most, a unique way of getting from (Δ_1) to (Δ) using spawnings and shufflings: spawning all the possible variables in any order and employing the only possible shuffle (using Proposition 4.1.3, axioms 1a, 1b, 1c); the order of spawning does not matter because shuffles are symmetric (Axioms 1c, 5c).

(2) Adding all of the objects that get linked and appear twice with different polarities on the premises, in any order, we obtain a new conclusion (Δ_2). There is a unique way of getting from (Δ_2) to (Δ_1) because of the interchanging axioms of linking (Axioms 5b and 5d). The only obstruction to uniqueness here is that we could choose different conclusions (Δ_2) depending on where they place the variables that will be linked; however, we have already shown that all possible choices lead to the same result (Proposition 4.1.6).

(3) We are left with a shuffle and some rushings and waitings. From the conclusion (Δ_2) we obtain now a sequence of premises $(\Gamma'_1, \dots, \Gamma'_n)$ that are the same as the original premises $(\Gamma_1, \dots, \Gamma_n)$ with the only difference that the objects appear ordered as in the conclusion (Δ_2). There is a unique possible shuffling from $(\Gamma'_1, \dots, \Gamma'_n)$ to (Δ_2).

(4) Finally, for each premise, there will be rushings and waitings. Rushings and waitings interchange (Proposition 4.1.5), and so there is a unique way of going from the original premise Γ_i to the new premise Γ'_i.

A summary of the proof can be found in Figure 7: Δ_1 is determined from Δ; Δ_2 is not, but all the possible choices lead to the same result (Figure 5); $\Gamma'_1, \dots, \Gamma'_n$ are determined from Δ_2 and $\Gamma_1, \dots, \Gamma_n$; we have argued that in each one of the steps of the proof there is a single possible derivation. □

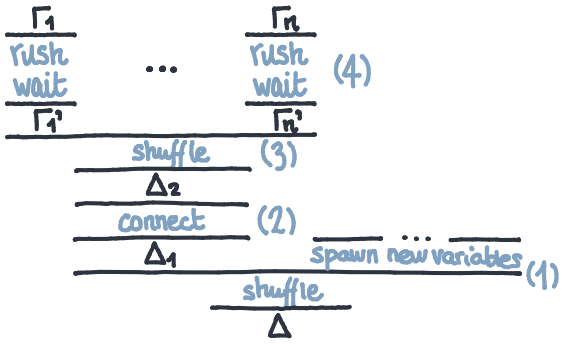


FIGURE 7. Schema for the proof of uniqueness.

Bibliography. Honda pioneered *binary session types* in the 90s [Hon93]; and further work with Yoshida and Carbone extended them to the multi-party case [HYC08]. Session types [Hon93, HYC08] are the mainstay type formalism for communication protocols, and they have been extensively applied to the π-calculus [SW01]. Our approach is not set up to capture all of the features of a fully-fledged

session type theory [KPT96]. Explicitly, our framework of message theories can be compared to asynchronous session types without choice. Arguably, this makes it more general: it always provides a universal way of implementing send (A°) and receive ($A^\bullet$) operations in an arbitrary process theory represented by a monoidal category. For instance, recursion and the internal/external choice duality [GH99, PS93] are not discussed, although they could be considered as extensions in the same way they are to monoidal categories: via trace [Has97] and linear distributivity [CS97b].

4.2. Physical Monoidal Multicategories, and Shufflings

Physical monoidal multicategories are physical duoidal categories [Spi13] where the parallel tensor ($\otimes$) is not representable. They follow the same idea of produoidal categories, but they will be a better framework for message passing. The theory of physical monoidal multicategories will need of three ingredients: *(i)* symmetric multicategories, which correspond to the symmetry of the parallel tensor ($\otimes$); *(ii)* monoidal multicategories, which correspond to the half-representable sequential tensor ($\triangleleft$); and *(iii)* normality, which we will need to reinterpret in this setting.

4.2.1. Symmetric Multicategories. Symmetric multicategories [BD98, CS09, Shu16] are to multicategories what symmetric monoidal categories are to monoidal categories. Let us write $\sigma \in S(n)$ for an element of the permutation group on n points. Let us write $\sigma_1 + \sigma_2 \in S(n+m)$ for the disjoint union of two permutations, $\sigma_1 \in S(n)$ and $\sigma_2 \in S(m)$. Let us write as $\sigma \wr (k_1, \dots, k_n) \colon k_{\sigma 1} + \ldots + k_{\sigma n} \to k_1 + \ldots + k_n$ the thickening of a permutation $\sigma \in S(n)$ to a permutation $S(k_1 + \cdots + k_n)$ that applies it *considering each block* of k_n elements separately.

Definition 4.2.1. A *symmetric multicategory* is a multicategory $\mathbb{M}$ together with the following family of functions

$$\sigma^* \colon \mathbb{M}(X_{\sigma 1}, \dots, X_{\sigma n}; Y) \to \mathbb{M}(X_1, \dots, X_n; Y), \text{ for each } \sigma \in S(n),$$

that moreover satisfy the following axioms: *(i)* functoriality, $(\tau \fatsemi \sigma)^*(f) = \tau^*(\sigma^*(f))$ and $\mathrm{id}^*(f) = f$; *(ii)* preservation of disjoint unions, $\sigma_1^*(f_1) \fatsemi_1 \ldots \fatsemi_{n-1} \sigma_n^*(f_n) \fatsemi_n g = (\sigma_1 + \ldots + \sigma_n)(f_1 \fatsemi_1 \ldots \fatsemi_{n-1} f_n \fatsemi_n g)$; and *(iii)* naturality, $f_1 \fatsemi_1 \ldots \fatsemi_{n-1} f_n \fatsemi_n \sigma^*(g) = (\sigma \wr (k_1, \dots, k_n))^*(f_{\sigma 1} \fatsemi_1 \ldots \fatsemi_{n-1} f_{\sigma n} \fatsemi_n g)$, for any f_n having arity k_n.

In physical monoidal multicategories, symmetry appears with the non representable parallel tensor ($\otimes$); while the sequential tensor will still form a representable monoidal category, this is the notion of monoidal multicategory.

4.2.2. Monoidal Multicategories. Multicategories helped us describe typed algebras: the objects of the multicategory were the types, and the multimorphisms were the operations we were allowed to use in that algebra. As we saw with produoidal categories, we need a second dimension if we want to study parallelism: what happens when the types themselves form a monoid? Monoidal multicategories are the monoidal version of algebra.

A *monoidal category* was a 2-monoid on the 2-category of categories, functors and natural transformations. Analogously, a *monoidal multicategory*, sometimes called a *virtual duoidal category* [Shu17] or *monoidal operad*, is a 2-monoid of the 2-category **Mult** of multicategories, multifunctors, and multinatural transformations.

Definition 4.2.2. A *monoidal multicategory* $(\mathbb{M}, \triangleleft, N, m, i, \fatsemi)$ is a multicategory $(\mathbb{M}, \fatsemi)$ with a tensor and a unit both on objects $(\triangleleft) \colon \mathbb{M}_{obj} \times \mathbb{M}_{obj} \to \mathbb{M}_{obj}$ and $N \colon \mathbb{M}_{obj}$, and a tensor and unit on multimorphisms,

$$m_n \colon \mathbb{M}(X_1, ..., X_n; Y) \times \mathbb{M}(X'_1, ..., X'_n; Y) \to \mathbb{M}(X_1 \triangleleft X'_1, ..., X_n \triangleleft X'_n; Y \triangleleft Y'),$$
$$n_n \colon 1 \to \mathbb{M}(N, \overset{n}{\dots}, N; N).$$

These are associative and unital up to multinatural transformations $\alpha_{X,Y,Z} \in \mathbb{M}(X \triangleleft (Y \triangleleft Z); (X \triangleleft Y) \triangleleft Z)$, $\lambda_X \in \mathbb{M}(I \triangleleft X; X)$ and $\rho_X \in \mathbb{M}(X \triangleleft I; X)$, satisfying the pentagon and triangle equations.

Remark 4.2.3. Any monoidal multicategory has an underlying monoidal category. Multifunctors between representable multicategories are lax monoidal functors, and multinatural transformations are lax monoidal transformations. This implies that a representable monoidal multicategory is exactly a duoidal category [Shu17].

4.2.3. Physical Monoidal Multicategories. Bringing the notion of normality to the context of monoidal multicategories is not completely trivial: here, there is no map between the two units $I \to N$. Instead, what we will have is a map between the hom-sets, given by the precomposition of the multimap $n_0 \in \mathbb{M}(;N)$ that we have because of the monoidality of the unit.

Definition 4.2.4. A *physical monoidal multicategory* is a symmetric multicategory $(\mathbb{M}, \fatsemi, \mathrm{id})$ that is moreover monoidal forming a virtual duoidal category $(\mathbb{M}, \triangleleft, N)$ and that contains an invertible composition with the unit map. That is, the following composition is an isomorphism,

$$(n_0 \fatsemi \bullet)^{-1} \colon \mathbb{M}(X_0, \overset{n}{\ldots}, X_n; Y) \to \mathbb{M}(X_0, \ldots, N, \ldots, X_n; Y).$$

Remark 4.2.5. We know the structure we need, but we will need to be able to compute with it. A better combinatorial description of physical monoidal multicategories will be possible once we characterize the free ones. The next section shows that the physical monoidal multicategory of shufflings is the free normal symmetric monoidal multicategory over a set of objects.

4.2.4. Shuffling. Shuffling events is the principle on which we have based our definition of message theories. The first step towards endowing them with categorical semantics is to show that shuffling arises naturally in a categorical setting: if physical duoidal categories represented inclusions of posets, physical monoidal multicategories will represent inclusions of linear posets, which correspond to shuffling.

Definition 4.2.6. Let A be an alphabet. The monoidal multicategory of *shuffling words*, wShuf_A, has as objects the set A^* of words on the alphabet A. The multimorphisms $\mathsf{wShuf}_A(w_1, \ldots, w_n; w)$ are precisely the shufflings of the letters of the words $w_1, \ldots, w_n$ that result into the word w. The monoidal tensor is given by concatenation of words.

Remark 4.2.7. This multicategory is not posetal: for instance, there exist two multimaps $\mathsf{wShuf}_A(a, a; aa)$ for any letter $a \in A$. More generally, the number of shuffles of a repeated letter is given by the binomial coefficients,

$$\#\mathsf{wShuf}(a^{n_1}, \ldots, a^{n_k}; a^{n_1+\ldots+n_k}) = \frac{(n_1 + \ldots + n_k)!}{n_1! \ldots n_k!}.$$

However, it is true that, if each letter appears at most once on the component words, then the shuffle, if it exists, must be unique. For instance, there exists a unique shuffle $\mathsf{wShuf}(xy, wz; xwyz)$, and so we can refer to it without explicitly specifying it. In other words, there exists at most one morphism between formal distinctly-typed expressions; shuffles are coherent.

Remark 4.2.8. This monoidal multicategory is symmetric and normal. In other words, it is a physical monoidal multicategory.

Proposition 4.2.9. *Shuffling words are the free physical monoidal multicategory on a set of objects.*

Proof. We have already shown how they form a physical monoidal multicategory; we need to show that it is the free one. Let $\mathbb{V}$ be any physical duoidal category, for each map $A \to \mathbb{V}_{obj}$ there must exist a unique physical monoidal multifunctor $\mathsf{wShuf}_A \to \mathbb{V}$ that factors on objects through the former map.

First, we argue that the physical monoidal multifunctor is forced. Any shuffle can be reconstructed from the operations of a physical monoidal multicategory, and so any shuffle must be mapped accordingly. For instance, if we want to reconstruct the shuffle $\mathbb{V}(xy, z; xzy)$, we multiply together: first $\mathbb{V}(x, i; x)$, then $\mathbb{V}(i, z; z)$, and finally $\mathbb{V}(y, i; y)$.

Let us do this in general. Assume a shuffle $\mathsf{wShuf}(w_1, \dots, w_n; w)$, where the words are $w_j = a_{j,1} \dots a_{j,k_j}$. We position each one of the letters in their position: for instance, if $a_{j,i}$ appears in the jth-word, we can use the normalization isomorphism

$$\mathbb{V}(a_{j,i}; a_{j,i}) \cong \mathbb{V}(I, \dots, a_{j,i}^{(j)}, \dots, I; a_{j,i})$$

to position it in the jth input, via the identity multimorphism. Then, we can multiply all of the letters forming the shuffle, using monoidality, and obtain the desired shuffle in any physical monoidal multicategory,

$$\prod_{a_{j,i} \in w} \mathbb{V}(I, \dots, a_{j,i}^{(j)}, \dots, I; a_{j,i}) \to \mathbb{V}(w_0, \dots, w_n; w).$$

We have used exactly one factor for each letter of the resulting word.

Finally, we need to prove that this assignment is well-defined and that it defines a physical monoidal multifunctor. These will be both a consequence of the same fact: there exists at most a unique shuffle between words with different letters and there exists at most a unique formally distinctly typed map between any poset shapes. To prove this, we use the characterization of the free duoidal category as poset shapes: there is a single formally well-typed map between any two poset shapes, so there is a single formally well-typed map between any two duoidal expressions; in particular, there exists at most one multimorphism between any words with distinct letters in a physical monoidal multicategory. This renders the map well-defined: there is at most one way of constructing any shuffle, and we have shown that it is possible. On the other hand, this also makes the map a physical monoidal multifunctor: all the equations are formal and thus they hold automatically. □

Shuffling takes care of the first and most important aspect of message theories: message theories are algebras for the shuffling words physical monoidal multicategory of their objects. However, message theories do have an extra structure: each object is a left dual, naturally with respect to these shufflings – the second ingredient for message theories is *polarization*.

4.2.5. Bibliography. This formulation of symmetric multicategories is a particular case of Shulman's definition of multicategory over a *faithful cartesian club* [Shu16, §2]. The theory of symmetric multicategories is less developed than the theory of *cartesian multicategories*, which are well-known to correspond to Lawvere theories. Monoidal multicategories are also not particularly explored, but Shulman has a note relating them directly to duoidal categories and proposing the name *virtual duoidal category* for them [Shu17].

4.3. Polarization

Polarization is not a property of our structures, but more of a particular way of constructing free structures. This is, we do not say that a monoid is "polarized"; but we have constructed free polarized monoids when talking about message theories. Saying that a monoid is polarized would seem to imply that all of its elements have a sign, which is not what happens in the free polarized monoid: generators do appear with a sign, but the words on the monoid are combinations without a particular sign.

Remark 4.3.1 (Polarization of a monoid)**.** Given any monoid M, we can consider its polarization, $\mathsf{Polar}(M)$, to be the monoid generated by two copies of each element of M, denoted $a^\bullet \in \mathsf{Polar}(M)$ and $a^\circ \in \mathsf{Polar}(M)$, and quotiented by the equations $a^\bullet b^\bullet = (ab)^\bullet$ and $a^\circ b^\circ = (ab)^\circ$.

Taking seriously this idea, we will not regard polarization as part of an algebraic structure (as it happens with monoidal categories). Instead, polarization will arise as a left adjoint. To quip, *polarization is left adjoint to taking left adjoints.* Let us see first how this works in the context of monoidal categories; we will later extend it to physical monoidal multicategories.

4.3.1. Monoidal Polarization. Every monoidal category has a notion of *duality* inside it. It does not suffice to say that some objects *have duals*: a duality is not only a property. Instead, we need to specify the maps that constitute the duality. Dualities in a monoidal category form a monoidal category themselves. Polarization is the left adjoint to taking this category of dualities.

Definition 4.3.2 (Category of dualities)**.** Let $(\mathbb{C}, \otimes, I)$ be a monoidal category. We define the category of left duals, $\mathsf{Duals}(\mathbb{C})$ to have objects the dualities $(L \dashv R, \varepsilon, \eta)$ of the monoidal category and morphisms

$$(f_L, f_R) \colon (L \dashv R, \varepsilon, \eta) \to (L' \dashv R', \varepsilon', \eta')$$

to be pairs of morphisms $f_L \colon L \to L'$ and $f_R \colon R' \to R$ such that the equations in Figure 8 hold.

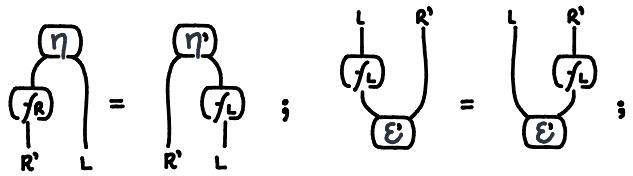


FIGURE 8. Morphism of adjunctions.

The category of left duals is a monoidal category where the tensor of two dualities, $(L \dashv R, \varepsilon, \eta)$ and $(L' \dashv R', \varepsilon', \eta')$, is defined to be

$$(L \otimes L' \dashv R' \otimes R, (\mathrm{id} \otimes \varepsilon \otimes \mathrm{id}) \fatsemi \varepsilon', \eta \fatsemi (\mathrm{id} \otimes \eta \otimes \mathrm{id})).$$

Taking the left duals extends to an endofunctor $\mathsf{Duals} \colon \mathbf{MonCat} \to \mathbf{MonCat}$. This endofunctor has a right adjoint $\mathsf{Polar} \colon \mathbf{MonCat} \to \mathbf{MonCat}$.

Definition 4.3.3 (Polarization)**.** Let $(\mathbb{C}, \otimes, I)$ be a strict monoidal category. Its polarization, $\mathsf{Polar}(\mathbb{C})$, is a monoidal category presented by the following data.

The polarization contains two objects, $A^\bullet$ and A°, for each object of the original category $A \in \mathbb{C}_{obj}$, and quotiented by equalities for tensors $(A \otimes B)^\bullet = A^\bullet \otimes B^\bullet$ and $(A \otimes B)^\circ = B^\circ \otimes A^\circ$ and units $I^\bullet = I^\circ = I$. It also contains a pair of morphisms

$\varepsilon_A \colon A^\bullet \otimes A^\circ \to I$ and $\eta_A \colon I \to A^\circ \otimes A^\bullet$ constructing an adjunction $A^\bullet \dashv A^\circ$: that is, $\varepsilon_A \fatsemi \eta_A = \mathrm{id}$ and $\eta_A \fatsemi \varepsilon_A = \mathrm{id}$. The duality is monoidal, satisfying both $\varepsilon_{A\otimes B} = \varepsilon_A \fatsemi \varepsilon_B$ and $\eta_{A\otimes B} = \eta_B \fatsemi \eta_A$, and finally $\varepsilon_I = \eta_I = \mathrm{id}_I$.

The polarization contains two dual morphisms, $f^\bullet \colon A^\bullet \to B^\bullet$ and $f^\circ \colon B^\circ \to A^\circ$, for each morphism $f \colon A \to B$. These are quotiented by equations explicitly asking for functoriality: $f^\bullet \fatsemi g^\bullet = (f \fatsemi g)^\bullet$ and $\mathrm{id}^\bullet = \mathrm{id}$, and also $f^\circ \fatsemi g^\circ = (g \fatsemi f)^\circ$ and $\mathrm{id}^\circ = \mathrm{id}$. These are tensored as $(f \otimes g)^\bullet = f^\bullet \otimes g^\bullet$ and $(f \otimes g)^\circ = g^\circ \otimes f^\circ$.

Proposition 4.3.4. *Polarization is left dual to taking left duals.*

Proof. We already know that $\mathsf{Duals} \colon \mathbf{MonCat} \to \mathbf{MonCat}$ is a functor; we only need to construct a universal arrow $\eta_{\mathbb{M}} \colon \mathbb{M} \to \mathsf{Duals}(\mathsf{Polar}(\mathbb{M}))$. The universal arrow will be given by $\eta(X) = (X^\bullet \dashv X^\circ, \varepsilon_X, \eta_X)$, which uses the fact that $X^\bullet \dashv X^\circ$ determines an adjunction. This assignment extends to a functor that sends a morphism $f \colon X \to Y$ to a morphism of dualities $(f^\bullet, f^\circ) \colon (X^\bullet \dashv X^\circ, \varepsilon, \eta) \to (Y^\bullet \dashv Y^\circ, \varepsilon, \eta)$, where $f^\circ \colon Y^\circ \to X^\circ$ is the dual of $f^\bullet \colon X^\bullet \to Y^\bullet$. This constructs the candidate universal arrow.

Let us show that it is indeed universal. Consider a functor $H \colon \mathbb{M} \to \mathsf{Duals}(\mathbb{N})$. We will show that there is a unique $H^* \colon \mathsf{Polar}(\mathbb{M}) \to \mathbb{N}$ such that $H = \eta_{\mathbb{M}} \fatsemi \mathsf{Duals}(H^*)$. Let us pick an object $X \in \mathbb{M}_{obj}$ that is sent to $H(X) = (A \dashv A', \varepsilon_A, \eta_A)$. To make this equation hold, it is necessary that $H^*(X^\circ) = A$ and $H^*(X^\bullet) = A'$, but also $H^*(\varepsilon_X) = \varepsilon_A$ and $H^*(\eta_X) = \eta_A$. Let us pick a morphism $f \colon A \to B$ that is sent to $H(f) = (g_L, g_R) \colon (A \dashv A', \varepsilon_A, \eta_A) \to (B \dashv B', \varepsilon_B, \eta_B)$; this forces $H^*(f^\bullet) = g_L$ and $H^*(f^\circ) = g_R$. Because $(A \dashv A', \varepsilon_A, \eta_A)$ forms a duality and the maps g_L and g_R are duals, this assignment satisfies all necessary equations. This determines the value of H^* on all of the generating maps while satisfying all of the equations of $\mathsf{Polar}(\mathbb{M})$. □

4.3.2. Monoidal Polarization is Not Enough. An important insight of some works into the categorical semantics of message passing is the importance of polarization. Given this, one could expect that the polarization of a process theory is its corresponding message theory. Indeed, this chapter will claim that the polarization of a symmetric monoidal category exhibits the necessary structure to discuss message passing; however, at the same time, it will claim that the algebraic structure that allows it to do so is not that of a polarized monoidal category: it is that of a message theory.

Remark 4.3.5. Some simpler interleavings of events can be expressed using dualities in a monoidal category. For instance, let us consider the polarization of a symmetric monoidal category: wires with the same sign can be swapped, but wires with different sign cannot. We can interleave two maps $f \colon I \to X^\circ \triangleleft Y^\circ \triangleleft Z^\bullet$ and $g \colon I \to U^\circ \triangleleft V^\bullet$ to produce a map $h \colon I \to X^\circ \triangleleft U^\circ \triangleleft V^\bullet \triangleleft Y^\circ \triangleleft Z^\bullet$ as in Figure 9 – note how we use the adjunction to contort the wires and allow wires to pass over wires with different polarization.

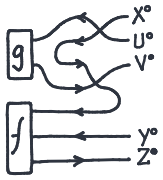


Figure 9. A shuffling, using polarization.

In general, thanks to the dualities, a *receiving* resource can be used later, $X^\circ \triangleleft A \to A \triangleleft X^\circ$, and a *sending* resource can be sent sooner, $A \triangleleft X^\bullet \to X^\bullet \triangleleft A$, but not the other way around. These two laws govern which wires can pass over other wires. A natural question, then, is to ask if all possible interleavings can be expressed in the same way: polarizing objects, and using dualities.

Proposition 4.3.6. *It is not the case that every shuffling of events can be expressed in a category with polarized objects and dualities.*

Proof. Consider two cells as in Figure 10, with types

$$f\colon I \to X^\bullet \triangleleft Y^\circ \triangleleft Z^\bullet \triangleleft W^\circ, \text{ and } g\colon I \to A^\bullet \triangleleft B^\circ \triangleleft C^\circ \triangleleft D^\bullet \triangleleft E^\bullet \triangleleft F^\circ$$

and imagine we want to interleave them into $A^\bullet \triangleleft B^\circ \triangleleft X^\bullet \triangleleft Y \circ \triangleleft C^\circ \triangleleft D^\bullet \triangleleft Z^\bullet \triangleleft W^\circ \triangleleft E^\bullet \triangleleft F^\circ$ using only the dualities. There are two possible cases: *(i)* we place f before g and we try to position the wires of f correctly; or *(ii)* we place g before f and we try to do the same.

In the first case, we will find that we can pass the wire X° over $A^\bullet \triangleleft B^\circ$ – but the same cannot be done with the wire $Y^\bullet$. Therefore, we are forced to pass the wire $Y^\bullet$ over $C^\circ \triangleleft D^\bullet \triangleleft E^\bullet \triangleleft F^\circ$, using the dualities; but then there will be no way of moving the wire $Z^\bullet$ to its place.

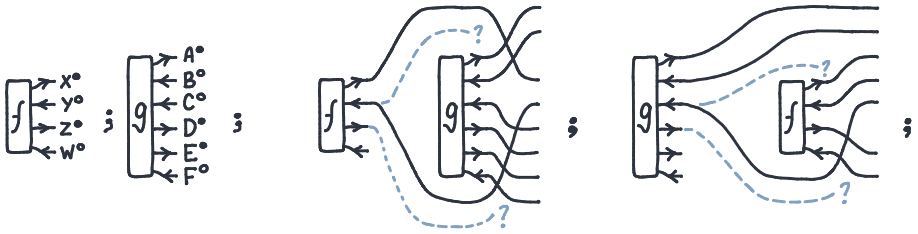


Figure 10. A shuffling that cannot be expressed.

In the second case, the first two wires, $A^\bullet \triangleleft B^\circ$, are automatically correctly placed. The third, C°, must pass over $Z^\bullet \triangleleft W^\circ$ because it could not do the same with $X^\bullet \triangleleft Y^\circ$; so $D^\bullet$ would be forced to do the same, but it cannot. □

Then, if this algebra is not what we are using about the polarization of a monoidal category, what are we using exactly? It happens that the polarization of a monoidal category posesses extra structure: its states (the morphisms without input) can always be split into morphisms that take negative objects on the left and produce positive objects on the right. This property is what makes it possible to express any interleaving of events.

With this in mind, the polarization of a monoidal category seems interesting not only because it produces a category with duals (or, in the work of Nester [Nes21], a proarrow equipment, or a *cornering*), but because it constructs a full message theory. We do not only care about the free polarization, we care about its message theory.

4.3.3. Polarization of a Physical Monoidal Multicategory. Every physical monoidal multicategory has a notion of dual, inherited from that of its underlying monoidal category. The next section will construct the free polarized physical monoidal multicategory over a set of objects. Let us define here the right adjoint: the functor that picks the set of left adjoints of a physical monoidal multicategory.

Definition 4.3.7. Let $(\mathbb{M}, \triangleleft, I)$ be a physical monoidal multicategory. We define the set of left duals, $\mathsf{Duals}(\mathbb{M})$, to have objects the dualities $(L \dashv R, \varepsilon, \eta)$ where $\varepsilon \in \mathbb{M}(L \triangleleft R; I)$ and $\eta \in \mathbb{M}(I; R \triangleleft L)$ are 1-to-1 multimorphisms satisfying the adjoint equations, $(\varepsilon \triangleleft \mathrm{id}_L) \fatsemi (\mathrm{id}_R \triangleleft \eta) = \mathrm{id}_L$ and $(\mathrm{id}_R \triangleleft \varepsilon) \fatsemi (\eta \triangleleft \mathrm{id}_R) = \mathrm{id}_R$.

4.3.4. Bibliography. A categorical treatment of polarization appears in the work of Cockett and Seely [CS07], which points to the connection with Abramsky-Jagadesaan games [AJ94]. "Polarized category" takes a different meaning there: it is a pair of categories endowed with a profunctor between them. However, we do follow the same core idea: using walking adjunctions for sending and receiving.

Nester notices the importance of polarization for message passing [Nes21] via a single-object proarrow equipment; and all the credit for this idea should go there. This was later extended by Nester and Voorneveld [NV23] to include iteration and choice. The reader will find a discussion of its relationship with lenses in a joint paper of this author with Nester and Boisseau [BNR22]. The graphical calculus of proarrow equipments was described by Myers [Mye16]; we can reuse this calculus to explain polarization in monoidal categories. Here, we prefer to avoid discussing polarization through proarrow equipments, noticing that this adjunction already works at the level of monoidal categories.

4.4. Polar Shuffles

4.4.1. Polar Shuffles. Polarization in a physical monoidal multicategory leads us to consider polar shuffles: shufflings endowed with a polarization for each one of its elements. This section will show that polar shuffles are the free polarized monoidal multicategory.

The objects of the category of shuffles could be seen as the finite ordinals: finite linear posets with inclusions on each other, preserving the ordering. Similarly, the objects of the category of polar shuffles are polarized finite ordinals.

Definition 4.4.1. A *polar list* over a set of types T is a list of types $X \in T^*$ endowed with a function $p\colon X \to \{\circ, \bullet\}$. In any polar list, we consider the *negative elements*, $X^\circ = p^{-1}(\circ)$, and the *positive elements*, $X^\bullet = p^{-1}(\bullet)$.

Definition 4.4.2. A *polar shuffle* over a set of types T, from a multiset of polar lists $X_0, \dots, X_n$ to a single polar list Y, is a bijection

$$f\colon X_1^\bullet + \dots + X_n^\bullet + Y^\circ \to Y^\bullet + X_1^\circ + \dots + X_n^\circ$$

preserving the types and such that the directed graph containing all polar lists (as linear posets) and an edge $x \to f(x)$ for each element in $X_1^\bullet + \dots + X_n^\bullet + Y^\circ$, is acyclic.

For instance, an untyped polar shuffle (or typed over the singleton set) of shape $\mathsf{pShuf}(\circ\bullet\circ\bullet, \circ\circ\bullet\bullet, \circ; \circ\circ\bullet\bullet\circ)$, is given by the following acyclic graph in Figure 11. In black, we depict the edges that come from the graph of a function. In blue, the edges that come from the linear finite posets.

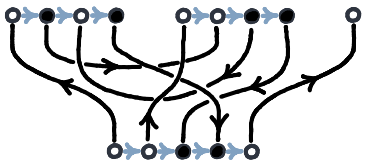
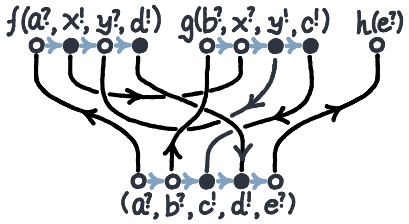


FIGURE 11. A polar shuffle.

4.4.2. Encoding of polar shuffles. Polar shuffles are ultimately graphs, and they can be encoded as such. We propose a notation suggestive of multiparty session calculi.

The *session encoding* of polar shuffles assigns a variable name to each polar list (say, $f, g, h, \dots$) and to each edge of the graph outside a polar list (say, $a, b, c, x, y, z, \dots$). The encoding of a polar shuffle starts by declaring the list of edges incident to the output polar list, together with their polarization. Then, enclosed in braces, we write the polar lists and the edges that incide on them.

For instance, the encoding of the polar shuffle in Figure 11 is in Figure 12.




```
(a?,b?,c!,d!,e?) {
  f(a?,x!,y?,d!),
  g(b,x?,y!,c!),
  h(e?) }
```

FIGURE 12. Encoding of a polar shuffle.

Remark 4.4.3. Parsing this notation requires checking whether the graph is acyclic. Checking if a graph is acyclic can be done in linear time on the sum of vertices and edges, $\mathcal{O}(v + e)$. The number of vertices in a polar shuffle is the sum of the lengths of all of the polar lists involved, $e = \#X_1 + \ldots + \#X_n + \#Y$, and each vertex receives at most three edges, giving a bound $e \leq 2v$. This means that checking if a polar shuffle is valid is linear in the length of its polar lists, $\mathcal{O}(\#X_1 + \ldots + \#X_n + \#Y)$.

The second implication of this encoding is that, if we label the vertices of a polar shuffle with types, there exists at most one polar shuffle with any distinctly typed sources and targets.

Proposition 4.4.4. *Polar shuffles are coherent. There exists at most a single polar shuffle between some distinctly typed polar lists: we say that a polar list is distinctly typed if each variable (each type) appears in the premises and the conclusion exactly twice, each time with a different variance.*

Proof. In this combinatorial structure, coherence works almost by definition. Note that a polar shuffle is ultimately a bijection satisfying certain extra properties; but the distinct typing already forces where each element must be sent: to the only element with different variance but same type. Whether the polar shuffle exists at all depends on whether it satisfies the acyclicity property. □

4.4.3. The Multicategory of Polar Shuffles. Polar shuffles form a multicategory, as their shape already suggests. The composition and the rest of the structure follow that of the category of shufflings. For instance, the composition of polar shuffles is defined to be the substitution of the resulting polar list of a shuffle into the input polar list of another shuffle. See Figure 13 for an example.

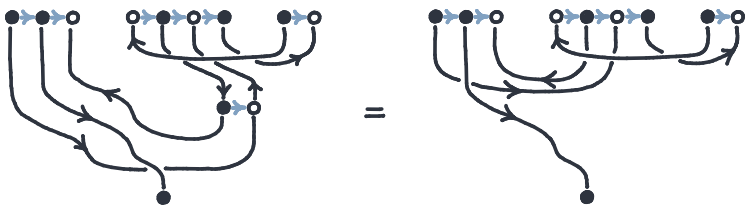


Figure 13. Composition of two polar shuffles.

Proposition 4.4.5. *The composition of two polar shuffles, $s \in \mathsf{pShuf}(X_1, \ldots, X_n; Y)$ and $t \in \mathsf{pShuf}(Y_1, \ldots, Y, \ldots, Y_m; Z)$, along a polar list Y, is a polar shuffle obtained by substituting the entire graph of the former into the polar list of the latter,*

$$s \wp_Y t \in \mathsf{pShuf}(Y_1, \ldots, X_1, \ldots, X_n, \ldots, Y_m; Z).$$

Substituting a polar shuffle into the inputs of a polar shuffle forms again a polar shuffle. That is, composition is well-defined and preserves acyclicity.

Proof. Composition happens across the resulting polar list of a polar shuffle, which coincides with one of the input polar lists of another polar shuffle; we say that this polar list is the *border*. The proof argues that two acyclic graphs glued along a linear graph are again acyclic.

Let us prove this by contradiction. Imagine there was a cycle in the multicategorical composition of polar shuffles. It must contain edges on both components of the composition, simply because each component is itself acyclic. This means that the cycle should cross the border between both components of the polar shuffle

at least twice and always an even number of times – it must take two different directions.

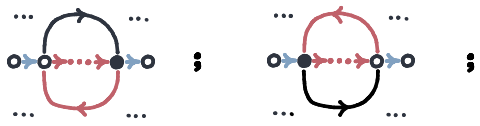


FIGURE 14. Composition along the borders of two polar shuffles.

Because the border is a linear poset, it must split the cycle (at least) in two parts, creating two undirected cycles to the two sides of the border. At least one of these two is forced to be directed, thus contradicting acyclicity on that side of the composition. □

Proposition 4.4.6. *The tensoring of two polar shuffles of the same arity,*

$$s \in \mathsf{pShuf}(X_0, \dots, X_n; Y) \text{ and } t \in \mathsf{pShuf}(X'_0, \dots, X'_n; Y'),$$

is a polar shuffle on the pointwise concatenation of the polar lists

$$(s \triangleleft t)\colon \mathsf{pShuf}(X_0X'_0, \dots, X_nX'_n; YY'),$$

defined by the disjoint union of the two acyclic graphs determining the polar shuffles. The tensoring of two polar shuffles is well-defined and it is again a polar shuffle. See Figure 15 for an example.

FIGURE 15. Parallel composition of polar shuffles.

PROOF. The graph of the tensoring is the disjoint union of two acyclic graphs, together with the edges determined by the polar lists. The directed edges coming from the polar lists always go from the first graph to the second; thus, they will not create cycles and the resulting graph with be acyclic. □

THEOREM 4.4.7. *Polar shuffles over a set of types are the morphisms of a physical monoidal multicategory,* pShuf*, that has the polar lists as objects.*

PROOF. We define the identity polar shuffle on a polar list to be the identity bijection linking each sign to itself. The identity polar shuffle is acyclic because the identity bijection preserves the linear ordering. We have already defined the composition and shown that it is acyclic in Proposition 4.4.5. Associativity of composition follows from associativity of glueing graphs; unitality follows by construction. We have already defined the tensoring in Proposition 4.4.6, and the tensor on objects is the concatenation of polar lists. The unit for the tensoring is the empty polar list, and because it does not appear in a polar shuffle, it makes the monoidal multicategory normal. □

4.4.4. Message Theories are Algebras of Polar Shuffles. What makes polar shuffles relevant to the discussion of message-passing is that they promise us a better syntax for message theories. Instead of thinking of the operations of a message theory as generated by a few primitives satisfying equations, we can give them a combinatorial characterization in terms of polar shuffles. This section shows that message theories are precisely the algebras for the operations given by polar shuffles.

Definition 4.4.8. An algebra over a multicategory, $(\mathbb{M}, ⅋, \mathrm{id})$, is the assignment of a set $S(X)$ to each object $X \in \mathbb{M}_{ob}$, and the assignment of a function

$$S(f)\colon S(X_1) \times \ldots \times S(X_n) \to S(Y),$$

for each multimorphism $f \in S(X_1, \ldots, X_n; Y)$. The assignment must preserve compositions, $S(f ⅋_X g) = S(f) ⅋_X S(g)$ and identities, $S(\mathrm{id}) = \mathrm{id}$. Alternatively, it is a multifunctor to the cartesian monoidal category of sets and functions.

Proposition 4.4.9. *Message theories are precisely the algebras of the free polarized physical monoidal multicategory over their respective sets of types. In other words, the derivations of a message theory form the free polarized physical monoidal multicategory over its set of types.*

Proof. This will follow almost by definition. The definition of message theories includes an operation for each shuffling, and these operations compose as shufflings (Proposition 4.1.3 and Axioms 1a, 1b, and 1c); equivalently, this is saying that message theories are in particular algebras of the multicategory of shufflings, the free physical monoidal multicategory over their set of types.

The rest of the axioms of message theories are exactly asking that each object is a left adjoint: the spawning and linking operations are describing the unit and the counit of the adjunction; the rest of the axioms are saying that: the unit of the adjunction is natural with respect to shufflings (Axioms 2a and 2b); the counit of the is natural with respect to shufflings (Axioms 3a and 3b); unit and counit satisfy the snake equations (4a and 4b); and they are natural with respect to each other (Axioms 5a, 5b, 5c and 5d). □

Corollary 4.4.10. *Polarized physical monoidal multicategories are coherent; there exists at most one multimorphism between any distinctly typed objects of the free physical monoidal multicategory over some objects.*

Proof. This is now a consequence of Proposition 4.4.9 and Theorem 4.1.8. The derivations of a message theory form the free polarized physical monoidal multicategory, but we have already shown that they are coherent. □

Theorem 4.4.11. *Polar shuffles form the free polarized physical monoidal multicategory over a set of types.*

Proof. Polar shuffles are coherent (Proposition 4.4.4), and polarized physical monoidal multicategories are coherent as well (Corollary 4.4.10); this simplifies the proof because, to show that they coincide, we only need to show that a polar shuffle between some types exists if an only if a multimorphism in the free polarized physical monoidal multicategory exists.

If a multimorphism of a certain type exists in the free polarized physical monoidal multicategory, then it exists in all polarized physical monoidal multicategories and, in particular, in polar shuffles (Theorem 4.4.7).

Let us prove the converse implication: if a polar shuffle between some types does exist, then there is a multimorphism in the free polarized physical monoidal multicategory between these types. For this, we will use that a polar shufflecan be

always factored (not necessarily uniquely!) in the following way: *(i)* we exchange positions inside each polar list to get them to their final relative position; *(ii)* we use a series of spawnings, or polar shuffles $I \to X^\circ \triangleleft X^\bullet$; *(iii)* we use a pure, non-polarized shuffle; and *(iv)* some final linkings of type $X^\bullet \triangleleft X^\circ \to I$. This is easy to verify topologically: we can always 'pull down the linkings' and 'pull up' the spawnings, and we can always factor any shuffle in two parts – a pure shuffle and a shuffle internal to each one of the components. For instance, consider the following example in Figure 16, adapted from Figure 15 with an extra spawning.

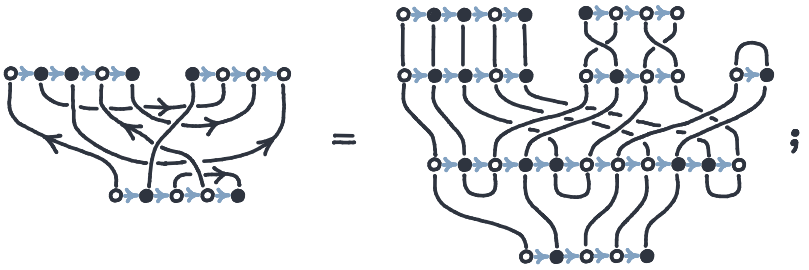


FIGURE 16. Factored polar shuffle.

Spawning, linking, shuffling, waiting and rushing are all operations on a physical monoidal multicategory (see Section 4.1); so this proves that there is at least one multimorphism in the free physical monoidal multicategory with these types. We do not care about the specific choice of multimorphism because polar shuffles are coherent (Proposition 4.4.4), and polarized physical monoidal multicategories are coherent as well (Corollary 4.4.10) □

Corollary 4.4.12. *Message theories are the algebras of the physical monoidal multicategory of polar shuffles.*

PROOF. We will use that polar shuffles are the free polarized physical monoidal multicategory (Theorem 4.4.11) and that message theories are precisely the algebras of the free polarized physical monoidal multicategory over their objects (Proposition 4.4.9). □

4.4.5. Bibliography. The definition – and the notation – of polar shuffles takes inspiration from a different concept: Hughes' *partial leaf functions* [Hug12]. Partial leaf functions are the **Int**-construction – the free compact closed category over a traced monoidal category – applied to the category of finite sets and partial functions [AM99]; here we follow a similar idea, but we work over finite sets and bijections, which are not traced. Not only the **Int**-construction, but also the idea of shuffling two traces point back to game semantics [MS18].

4.5. Processes versus Sessions

4.5.1. Processes of a message theory. Inside of a message theory, we call *processes* the sessions that happen in two parts: *(i)* first they ask for some indeterminate number of resources (possibly zero), $X_1^\circ, \dots X_n^\circ$ and *(ii)* then, they give some indeterminate number of resources (possibly zero), $Y_1^\bullet, \dots, Y_m^\bullet$. This simple definition builds a symmetric monoidal category inside any message theory, and we will find a left adjoint to this construction.

Proposition 4.5.1. *Let $\mathbb{M}$ be a message theory. There exists a symmetric monoidal category,* $\mathsf{Proc}(\mathbb{M})$*, whose objects are the lists of objects of the message theory,* $\mathsf{Proc}(\mathbb{M})_{obj} = \mathbb{M}^*_{obj}$*, and whose morphisms are the sessions that first ask some resources and then provide some resources,*

$$\mathsf{Proc}(\mathbb{M})(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m) = \mathbb{M}(X_n^?, \dots X_1^?; Y_1^!, \dots, Y_m^!).$$

Note how we reverse the order of inputs; this will make reasoning easier even if it is unnecessary: the monoidal category will be symmetric in any case.

Proof. Let us define the composition and identities. Composition is given by the polar shuffle that connects the middle outputs to inputs, see Figure 17; because message theories are algebras of polar shuffles, such a polar shuffle defines an operation, composition, that takes two sessions to a third one.

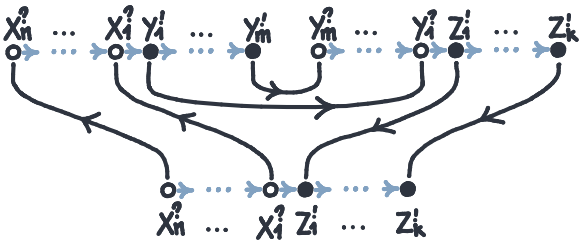


Figure 17. Composition of processes of a message theory.

Identities are created by spawning channels, see Figure 18. Again, because message theories are algebras of polar shuffles, each message theory must contain a constant given by the polar shuffle that spawns a list of channels. Composition and identities are associative and unital: it can be checked from the definition that we are using the duality from spawning and connecting channels.

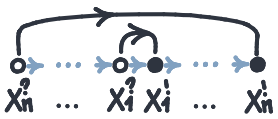


Figure 18. Identity process of a message theory.

Symmetries are given by the polar graph that spawns a channel for each one of the objects and then shuffles them so that the inputs and the outputs are divided in two blocks and positioned in reverse order, see Figure 23.

Tensoring is given by the polar shuffle that preserves all outputs and inputs but passes the outputs of a process pass the inputs of the other, see Figure 20.

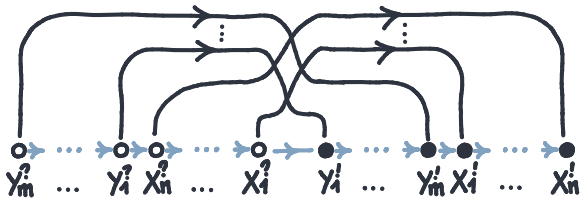


FIGURE 19. Symmetries of processes in a message theory.

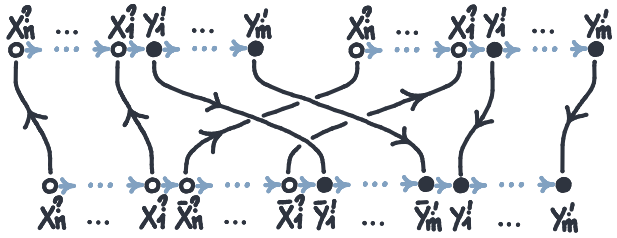


FIGURE 20. Tensor of processes in a message theory.

This operation is again associative and unital with the empty polar shuffle that represents the monoidal unit.

It concludes the proof to check, by following the connections of the polar shuffles, that tensoring behaves functorially with composition so that the interchange law of monoidal categories holds. □

Proposition 4.5.2. *The construction of the symmetric monoidal category of processes extends to a functor*

$$\mathsf{Proc}\colon \mathbf{Msg} \to \mathbf{SymMonCat}_{\mathsf{Str}}.$$

PROOF. We will show that any message functor $F\colon \mathbb{M} \to \mathbb{N}$ induces a strict symmetric monoidal functor $\mathsf{Proc}(F)\colon \mathsf{Proc}(\mathbb{M}) \to \mathsf{Proc}(\mathbb{N})$. Because the category of processes is freely generated on objects, it suffices to explain that the object $X \in \mathbb{M}_{ob}$ is sent to the object $FX \in N_{ob}$. On morphisms, we already have a map

$$F\colon \mathbb{M}(X_n^?, \dots, X_1^?, Y_1^!, \dots, Y_m^!) \to \mathbb{N}(FX_n^?, \dots, FX_1^?, FY_1^!, \dots, FY_m^!)$$

that gets reinterpreted as a map

$$\mathsf{Proc}(\mathbb{M})(X_1 \otimes \dots \otimes X_n; Y_1 \otimes \dots \otimes Y_m) \to \mathsf{Proc}(\mathbb{N})(F(X_1 \otimes \dots \otimes X_n); F(Y_1 \otimes \dots \otimes Y_m)).$$

Composition, identities and tensoring are operations constructed as polar shuffles, and so they must be preserved by a message functor; this means that $\mathsf{Proc}(F)$ becomes a strict monoidal functor. □

4.5.2. Sessions of a process theory. We will now construct message sessions over an arbitrary process theory, and we will do so in a minimalistic theory. Message passing consists of two effects: *sending* and *receiving*. Premonoidal categories already are a framework for effectful computation in process theories, so we employ them here.

Definition 4.5.3. The effectful category of *sessions* over a strict symmetric monoidal category $\mathbb{C}$ is the effectful category $\mathbb{C} \to \mathsf{Session}(\mathbb{C})$ generated by

(1) all of the morphisms of the original monoidal category, $\mathbb{C}$, quotiented by the equations of the original monoidal category, as pure morphisms;
(2) and a pair of *send* and *receive* generators for each object $X \in \mathbb{C}_{obj}$ imposing no further equations. We write these generators as $(\circ)_X : X \to I$ and $(\bullet)_X : I \to X$.

FIGURE 21. Session runtime generators.

This naive theory of message passing on top of a monoidal category may remind us of the combs that we were studying before: instead of using incomplete diagrams, we are marking the interchanges explicitly now. This intuition can be made formal: we will now prove that combs of type $X_1^\bullet \triangleleft \ldots \triangleleft X_n^\bullet \to \binom{A}{B}$ correspond to sessions $A \to B$ where the events are exactly $X_1^\bullet, \ldots, X_n^\bullet$, and happen in that specific order. First, note that we can define a sequence of events associated with a particular session.

Definition 4.5.4. The *sequence of events* of a session is the list of effectful generators obtained by following only the effectful wire on the diagram. Formally, it is defined by structural induction over the premonoidal category of sessions as follows.

(1) It is the empty list for a pure morphism.
(2) It is invariant to whiskering.
(3) It contains a single element $[X^\circ]$ for each generator $(\circ)_X : I \to X$.
(4) It contains a single element $[X^\bullet]$ for each generator $(\bullet)_X : X \to I$.
(5) It concatenates the lists for a composition of morphisms.

It becomes straightforward to check that the sequence of events is well-defined. We write $\mathsf{Session}(A;B)[X_1^\bullet, \ldots, X_n^\bullet]$ for the set of sessions $A \to B$ with a sequence of events $X_1^\bullet, \ldots, X_n^\bullet$.

Proposition 4.5.5 (Combs are sessions)**.** *Sessions from A to B with a sequence of events $X_1^\bullet, \ldots, X_n^\bullet$ are in bijective correspondence with combs with the same events happening sequentially,*

$$\mathsf{Session}(A;B)[X_1^\bullet, \ldots, X_n^\bullet] \cong \mathsf{mLens}\left(X_1^\bullet \triangleleft \ldots \triangleleft X_n^\bullet; \tbinom{A}{B}\right).$$

PROOF. We proceed by structural induction over the presentation of the cteogry of sessions. The base case consists of a morphism that is pure: by definition, those are the combs $\mathsf{mLens}(\mathsf{N}; \binom{A}{B})$.

The inductive case considers a morphism $A \to B$ with at least one first occurrence of the effectful generators, $(\circ)_X$ or $(\bullet)_X$. We assume without loss of generality that this is $(\bullet)_X : I \to X$, so its sequence of events is $X^\bullet, \Gamma$ – the other case is analogous. We consider it as a string diagram for a monoidal category, quotiented by the equations of the base monoidal category $\mathbb{C}$, the symmetries and only up to interchange by isotopy. We may split the diagram into two parts (as in Figure 22); we leave everything before the generator $(\bullet)_X$ to one side, $f \in \mathbb{C}(A; X \otimes M)$, and everything after the generator to the other side $g \in \mathsf{Session}(M, B)[\Gamma]$. This procedure constructs the following comb,

$$\int^{M \in \mathbb{C}} \mathbb{C}(A; M \otimes X)_f \times \mathsf{Session}(A;B)[\Gamma]_g.$$

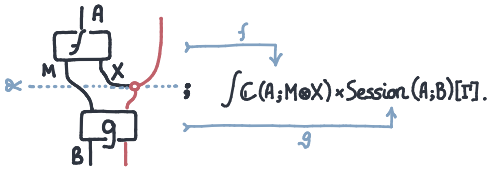


FIGURE 22. Splitting the diagram of a session.

Of course, the usual problem with this kind of definitions that split a string diagram is that we need to prove that they are well-defined. We need to show that this definition is invariant to isotopy; we do so by cases: *(i)* if the isotopy interchanges two boxes before the cutting line, then there were two pure morphisms and it is captured by an equation of $\mathbb{C}$; *(ii)* if the isotopy interchanges two boxes after the cutting line, then it is defining an equation of sessions; *(iii)* if the isotopy interchanges two morphisms across the cutting line, then it is captured by dinaturality.

At the same time, note that these are exactly all of the equations imposed to combs: those of the original symmetric monoidal category and dinaturality; so the correspondence is bijective. Finally, by the induction hypothesis,

$$\begin{aligned}\mathsf{Session}(A;B)[X^\bullet,\Gamma] &\cong \int^{M\in\mathbb{C}} \mathbb{C}(A;X\otimes M)\times \mathsf{Session}(A;B)[\Gamma]\\ &\cong \int^{M\in\mathbb{C}} \mathbb{C}(A;X\otimes M)\times \mathsf{mLens}\left(\Gamma;\tfrac{A}{B}\right)\\ &\cong \mathsf{mLens}\left(X^\bullet\triangleleft\Gamma;\tfrac{A}{B}\right),\end{aligned}$$

which concludes the proof. □

The next step is to show that sessions over a process theory actually define a message theory. For this, we will only need sessions with no inputs or outputs: we write $\mathsf{Session}[\Gamma]$ for $\mathsf{Session}(I;I)[\Gamma]$.

Proposition 4.5.6. *Sessions over a strict symmetric monoidal category,* $\mathsf{Session}(\mathbb{C})$*, form a message theory. This construction extends to a functorial assignemnt*

$$\mathsf{Session}\colon \mathbf{SymMonCat}_{\mathsf{Str}} \to \mathsf{Msg}.$$

PROOF. We will show that sessions over a strict symmetric monoidal category, $\mathsf{Session}(\mathbb{C})$, form an algebra for the multicategory of polar shuffles. Consider a family of sessions, $s_i \in \mathsf{Session}(\Gamma_i)$; and consider a polar shuffle

$$p \in \mathsf{pShuf}(\Gamma_1,\dots,\Gamma_n;\Delta),$$

we need to construct a new session of type Δ.

The construction follows a topological intuition: consider the hypergraph defining the string diagrams of the sessions; and consider at the same time the acyclic graph defined by the polar shuffle. We glue the string diagram of each s_i, along its runtime wire, to the inputs and outputs, Γ_i, of the polar shuffle, removing these nodes on the process. The crucial step happens now: we have a graph containing nodes of the premonoidal category of sessions, and it has been constructed by gluing acyclic graphs along linear boundaries – it must be acyclic, and it must be a string diagram.

We define the composition of the sessions $s_i \in \mathsf{Session}(\Gamma_i)$ along the polar shuffle $p \in \mathsf{pShuf}(\Gamma_1,\dots,\Gamma_n;\Delta)$, to be the session represented by the string diagram here obtained. This assignment preserves the composition of polar shuffles – which is also defined topologically by gluing – and thus it determines an algebra.

Finally, the assignment is functorial with respect to the base monoidal category because *(i)* the construction of the sessions is functorial and *(ii)* we only apply

operations of a symmetric premonoidal category when we build a string diagram over this premonoidal category of sessions. □

Example 4.5.7. Consider two morphisms, $f\colon A \to M \otimes X$ and $g\colon M \otimes Y \to B$, determining a session of type $[A^\circ, X^\bullet, Y^\circ, B^\bullet]$; and consider two other morphisms, $h\colon X \otimes U \to N$ and $k\colon N \to Y \otimes V$, determining a session of type $[X^\circ, U^\circ, V^\bullet, Y^\bullet]$. They can compose along the polar shuffle we defined in Figure 12; the result is in Figure 23.

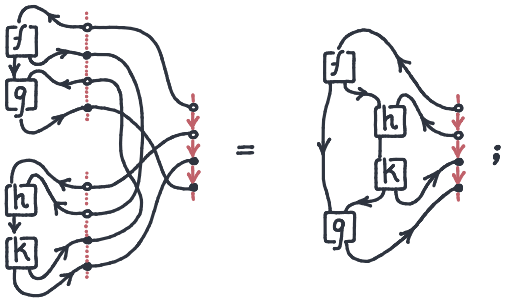


FIGURE 23. Two sessions compose along a polar shuffle.

4.5.3. Sessions versus Processes. The final result of this section is to prove that *sessions* over a process theory are the free message theory over a symmetric monoidal category.

Lemma 4.5.8. *There exists a strict symmetric monoidal functor*

$$\mathsf{inProc}\colon \mathbb{C} \to \mathsf{Proc}(\mathsf{Session}(\mathbb{C}))$$

that includes a monoidal category in the processes of its message theory.

PROOF. The functor will act as the identity on objects. We already know that combs are sessions (Proposition 4.5.5), and we can use this fact to construct the assignment on morphisms.

$$\mathsf{Proc}(\mathsf{Session}(\mathbb{C})(A;B)) = \mathsf{Session}(\mathbb{C})[A^\circ, B^\bullet] \cong \mathsf{mLens}(A^\circ, B^\bullet; {}^{I}_{I}) \cong \mathbb{C}(A;B).$$

It only remains to show that this assignment defines a strict symmetric monoidal functor: we need to show that it preserves composition, tensoring, identities and symmetries. This is straightforward, as we only need to check that the operations that we defined for the process theory of a message theory, $\mathsf{Proc}(\mathbb{M})$, correspond to the operations of a symmetric monoidal category. Let us explicitly check composition, $\mathsf{inProc}(f) \fatsemi \mathsf{inProc}(g) = \mathsf{inProc}(f \fatsemi g)$, in Figure 24, the rest follow a similar pattern.

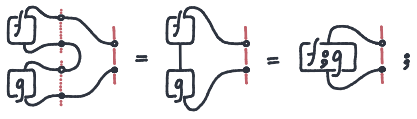


FIGURE 24. The inclusion of processes preserves composition.

Checking the rest of the cases concludes the construction. □

THEOREM 4.5.9. *Sessions and processes form an adjunction,* $\mathsf{Session} \dashv \mathsf{Proc}$*; where sessions,* $\mathsf{Session}\colon \mathbf{SymMonCat}_{\mathsf{Str}} \to \mathsf{Msg}$*, construct the free message theory over a symmetric monoidal category, and where processes,* $\mathsf{Proc}\colon \mathsf{Msg} \to \mathbf{SymMonCat}_{\mathsf{Str}}$*, construct the cofree symmetric monoidal category over a message theory.*

PROOF. Consider a strict symmetric monoidal category, $\mathbb{C}$, and a message theory, $\mathbb{M}$, endowed with a strict symmetric monoidal functor $F\colon \mathbb{C} \to \mathsf{Proc}(\mathbb{M})$. We will construct a message functor $F^\sharp\colon \mathsf{mLens}(\mathbb{C}) \to \mathbb{M}$ and prove that it is the unique one satisfying $\mathsf{inProc} \fatsemi \mathsf{Proc}(F^\sharp) = F$.

Let us show that such a message functor, if it were to exist, would be unique. Firstly, the image on message types is already determined to be $F^\sharp(A) = F(A)$. Secondly, the image on sessions consisting on a single morphism, $\mathsf{inProc}(f)\colon [A^\circ, B^\bullet]$, is determined, $F^\sharp(\mathsf{inProc}(f)) = F(f)\colon [FA^\circ, FB^\bullet]$. We will show now that this reasoning can be extended to all sessions: we know that sessions of type $[X_1^\bullet, \dots, X_n^\bullet]$ are combs (Proposition 4.5.5) of type

$$(f_0|\dots|f_n)\colon X_1^\bullet \triangleleft \dots \triangleleft X_n^\bullet \to (I^I)\,.$$

This determines their image: combs can be factored as the composition, in the message theory of multiple sessions consisting of a single morphism (Figure 25). Accordingly, their image, $F^\sharp(f_0|\dots|f_n)$, should be the composition, on the message theory, of these pieces (Figure 25).

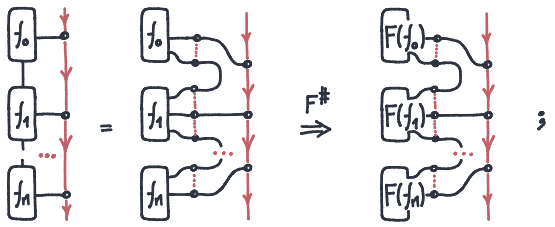


FIGURE 25. Image of a comb under the message functor.

Let us now show that we have constructed a well-defined assignment. Our construction should preserve the dinaturality equivalence relation imposed to combs. This happens, indeed, and the proof simply checks that the images of two combs,

$$(f_0 \fatsemi (\mathrm{id} \otimes h_0)|f_1 \fatsemi (\mathrm{id} \otimes h_1)|\dots|f_n) = (f_0|(\mathrm{id} \otimes h_0) \fatsemi f_1|\dots|(\mathrm{id} \otimes h_{n-1}) \fatsemi f_n),$$

are equal (Figure 26). We have constructed a well-defined assignment on sessions.

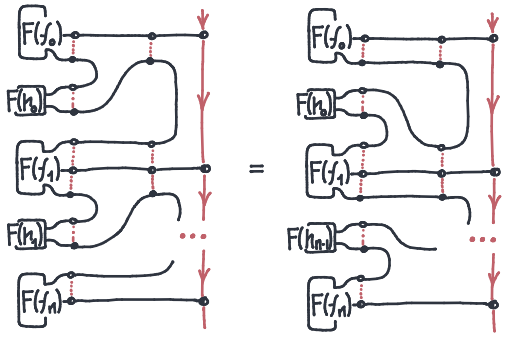


FIGURE 26. The assignment of combs to sessions preserves dinaturality.

Finally, we need to check that $F^\sharp$ is a message functor preserving all of the operations determined by polar shuffles. In the theory of combs, applying a polar shuffle corresponds to a rewiring into another comb; applying the polar shuffle in $\mathbb{M}$ must

result in the same rewiring of the pieces forming the comb – which, as we have already shown, is precisely the image of that comb. This forces $F^\sharp$ to preserve the application of a polar shuffle.

We have shown that $F^\sharp$ is indeed a message functor and that it is the only possible one satisfying $\mathsf{inProc} \fatsemi \mathsf{Proc}(F^\sharp) = F$. □

4.5.4. Example: One-Time Pad, as a Message Session. Let us come back to Example 3.5.8, where we discussed a decomposition of the one-time pad. We now know that there is an adjunction between symmetric monoidal categories and message theories, let us use it to provide semantics to the decomposition of the one-time pad example.

The theory for the one-time pad problem can be expressed in a message theory $\mathbb{O}$ where we have a single object generator for the type of a message, X, and a single session generator for each one of the actors.

(1) Stage: $X^\circ \triangleleft X^\bullet \triangleleft X^\bullet$,
(2) Bob: $X^\bullet \triangleleft X^\circ \triangleleft X^\bullet$,
(3) Alice: $X^\circ \triangleleft X^\circ \triangleleft X^\bullet$,
(4) Eve: $X^\circ \triangleleft X^\bullet$.

We will interpret these generators in the free message theory over the category of finite sets and stochastic maps: thanks to the adjunction, we know that, if we could interpret each one of the components presenting the category of finite sets in any message theory, then we can interpret this whole example inside it.

We already gave an interpretation to each one of the components in terms of combs. We rewrite now the example explicitly separating each one of the parts that form it (Figure 27).

Proposition 4.5.10. *The session describing the one-time pad protocol is equal to a session where* Alice *and* Bob *communicate the message directly and* Eve *attacks a signal representing pure noise.*

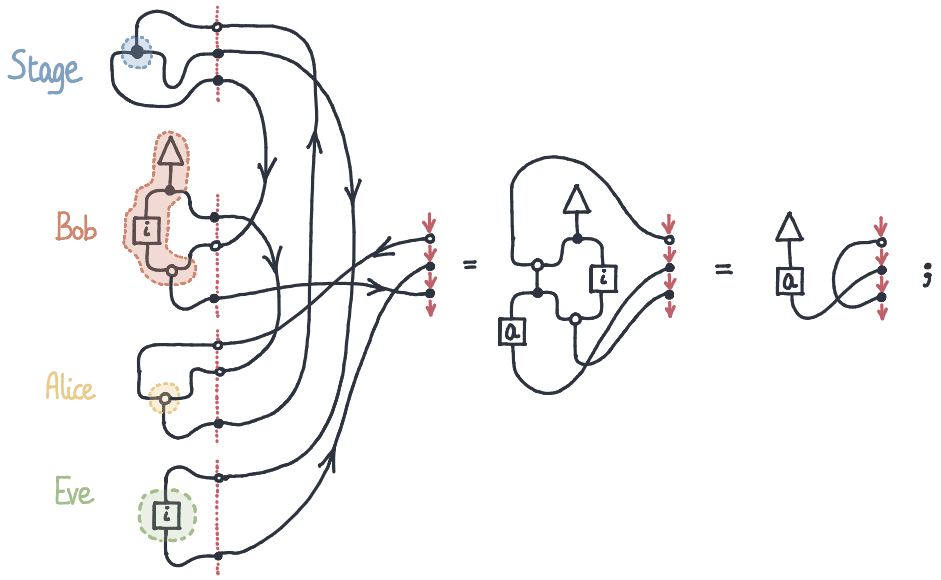


FIGURE 27. One-time pad, complete session.

PROOF. Evaluating the session that describes the one-time pad example using the components described before, in Example 3.5.8, obtains the following polar shuffle applied to multiple combs. Evaluating the polar shuffle, as in Figure 27, produces the desired result. □

Remark 4.5.11. This discussion is not restricted to the modularity of the string diagrams: it affects the modularity of the code itself. Recall that we have a notation for sessions and polar shuffles; we use it in Figure 28 to write the one-time pad.

```
oneTimePad(?msg, !crypt, !decrypt) = {
  bob(!key, ?cryptBob, !decrypt),
  alice(!msg, ?key),
  eve(!cryptEve),
  stage(?crypt, !cryptBob, !cryptEve),
}
```

FIGURE 28. Notation for the one-time pad session.

At the same time, we have second a notation for sessions: sessions are ultimately morphisms of an effectful category, so we can use *do-notation* without the interchange axiom to represent them. The sending and receiving effects can be written as (!/?) respectively. Lenses are tuples of morphisms, and they can be represented in do-notation using that exact characterization. The following Figure 29 shows a modular implementation of the one-time pad that separates each one of the actors into a different module.

```
oneTimePad(alice,bob,eve,msg) = do
  key <- bob₀()
  crypt <- alice(msg, key)
  () <- eve(crypt)
  msg <- bob₁(crypt)
  return msg
```

```
alice(msg, key) = do
  crypt <- xor(msg,key)
  return crypt
```

```
bob() = do
  key <- randomBit
  !key
  ?crypt
  msg <- xor(crypt,key)
  return msg
```

```
eve(crypt) = do
  return crypt
```

FIGURE 29. Do-notation for the one-time pad.

Which notation should we settle for? It seems that both interplay nicely together: the best way of writing a message session seems to be to write its underlying polar shuffle, as in Figure 28, while the best way of writing processes may be the usage of do-notation as in Figure 29, which is well-known and imposes a human-readable order on the operations.

4.5.5. Case Study: Causal versus Evidential Decision Theories. Leibniz's dream was to see philosophical debates reduced to mathematical calculation, to have a formal language for decision theory and an algorithm to solve any dispute.

> "[...] if controversies were to arise, there would be no more need of disputation between two philosophers than between two calculators. For it would suffice for them to take their pencils in their hands and to sit down at the abacus, and say to each other (and if they so wish also to a friend called to help): *calculemus* (let us calculate)."

However, our modern decision theory seems far from this dream. For instance, Monty Hall's problem caused famous controversies and confident blunders of some experts [vS] while being relatively simple to describe. It could seem that the passage from the statement to its formal encoding is more of an art than a science.

Let us try to understand one of these debates: *causal* versus *evidential* decision theory on Newcomb's problem [Noz69, Ahm14, YS17]. We will use message theories to set up the scene and partial Markov categories to compute the solution.

Definition 4.5.12. Newcomb's problem [Noz69] is a famous decision problem that sets apart Evidential and Causal Decision Theory. An agent (🧍) is in front of two boxes: a transparent box filled with 1€ and an opaque box (📦📦). The agent is given the choice between taking both boxes (two-boxing, T) and taking just the opaque box (one-boxing, O). However, the opaque box is controlled by a "perfectly accurate" predictor (🔺). The predictor placed 1000€ in the opaque box if it predicted that the agent would one-box and left it empty otherwise. The agent knows this. Which action should the agent choose?

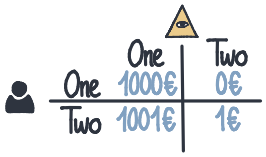


FIGURE 30. Newcomb's problem: table of utilities.

At the risk of oversimplifying, most philosophers are divided in two schools [Ahm14, YS17]. Those that follow *causal decision theory* would claim that no matter what the predictor does, the lower row of the table in Figure 30 contains strictly more utility; they prescribe *two-boxing*. Those that follow *evidential decision theory* claim that, because the predictor is omniscient, *one-boxing* is the only way of ensuring the biggest prize is on the box.

The analysis of the problem starts by dividing it into different parties: *(i)* the agent (🧍) must only make a choice on whether to one-box or two-box; *(ii)* the stage (📦📦) takes the choice of the agent, the prediction of the predictor, broadcasts the choice of the agent and computes the final utility of the agent, and *(iii)* the predictor (🔺) sends a prediction and, only afterwards, can see the choice of the agent. Let us call $X = \{\mathsf{O}, \mathsf{T}\}$ to the set containing *one-boxing* or *two-boxing*; we are claiming to have three elements of a message theory: the agent, (🧍)$\colon X^\bullet$; the predictor, (🔺)$\colon X^\bullet \triangleleft X^\circ$; and the stage, (📦📦)$\colon X^\circ \triangleleft X^\circ \triangleleft X^\bullet \triangleleft X^\bullet$.

FIGURE 31. Newcomb's problem: components of a message theory.

The Evidential Decision Theory Solution: Let us take as an axiom that these components are constructed out of total stochastic channels; in other words, the message theory we use is the free message theory over the Kleisli category of the subdistribution monad, $\mathsf{Session}(\mathsf{Kleisli}(\mathbf{D}_{\leq 1}))$.

We cannot assume anything about the agent, but because of the construction of out free message theory (Definition 4.5.3), it must be given by a single stochastic channel $(\blacktriangle)\colon I \to X$. Even without assuming anything about the predictor, because of the construction of the free message theory (Definition 4.5.3), we know that must be constructed of two parts: the one that sends the prediction, $(\blacktriangle)_1\colon I \to M \otimes X$, and the one that receives the choice of the agent, $(\blacktriangle)_2\colon M \otimes X \to I$; of the first part we know nothing, but we have postulated that we will observe it to be perfectly accurate with the prediction, meaning that the second part will fail if it is not. Thus, we deduce it must factor as in Figure 32.

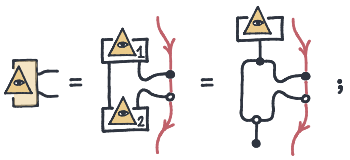


FIGURE 32. Evidential reading of the predictor.

The wiring of the components is given by the statement: agent and predictor send choice and prediction to the stage, which answers giving back the choice to the predictor and computing the output (Figure 35). We then reason *(i)* computing the polar shuffle; *(ii)* we analyze the agent by cases, the agent two-boxes (T) with probability a or one-boxes (O) with probability $(1-a)$; *(iii)* because both cases are deterministic, they can be copied; *(iv)* we analyze then the predictor, it two-boxes (T) with probability p or one-boxes (O) with probability $(1-p)$; *(v)* we compute according to Figure 30, canceling the incompatible equality checks; and *vi* assuming that the first term is just an order of magnitude larger than the second, we can bound it by $p \cdot 1000€$, where we take $a = 1$: the agent should one-box.

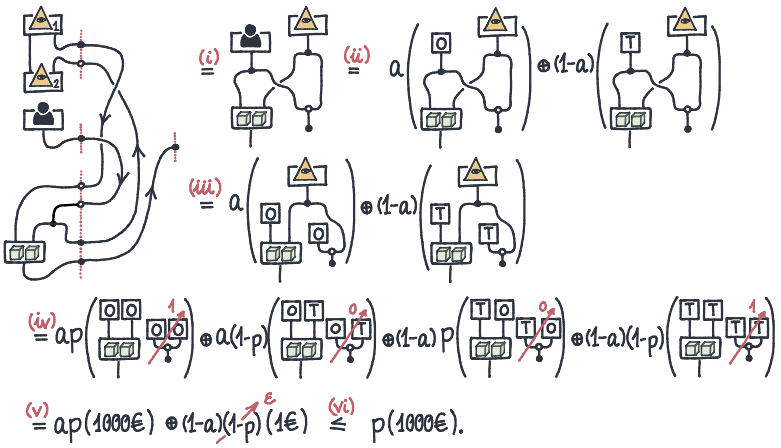


FIGURE 33. Newcomb's problem: the solution from Evidential Decision Theory.

The Causal Decision Theory Solution: Let us assume the same components (Figure 30). The only hypothesis over which we will place suspicion is that the predictor can be "perfectly accurate" without violating causality in some way. Causal decision theory assumes that all processes are causal, or total. That means that, after receiving the news of what the agent has chosen, the predictor can do nothing: there must exist a unique total morphism $X \otimes X \to I$.

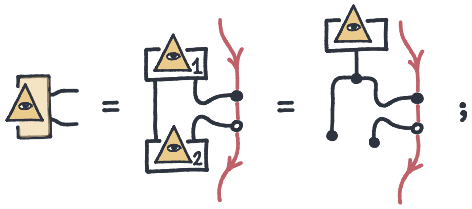


FIGURE 34. Causal reading of the predictor.

The wiring of the components is again the same, but the computation is now different: *(i)* we compute the polar shuffle; *(ii)* we analyze the agent by cases, the agent two-boxes (T) with probability a or one-boxes (O) with probability $(1-a)$; *(iii)* we analyze in the same way the predictor; *(iv)* we compute according to Figure 30, canceling the incompatible equality checks; and *(v)* we bound everything by the case where $a = 0$: this time, to maximize utility, the agent must two-box.

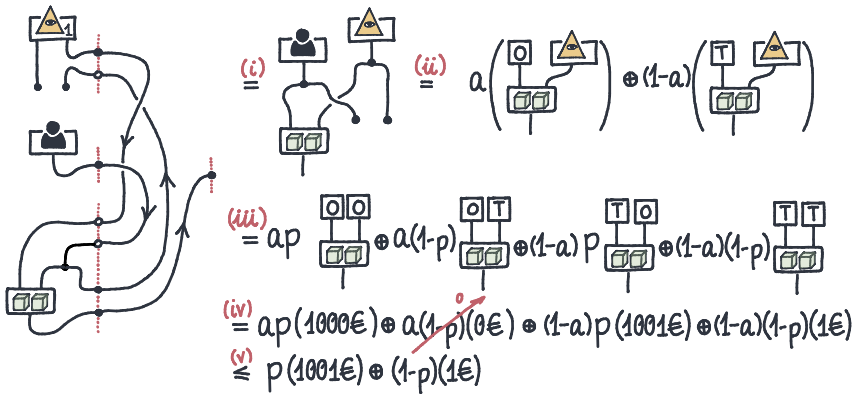


FIGURE 35. Newcomb's problem: the solution from Causal Decision Theory.

Was this formal analysis better than a pure discussion? We can now claim that the advantage is that the computations go from a starting diagram that represents our reading of the problem to a utility that we can maximize. We have turned most of the problem into a problem of computation: not only looking at the table of utilities (Figure 30) but at the whole statement of the problem. Of course, our formalization does not solve the debate on Newcomb's paradox, but at least it moves the controversy to a more fundamental point: are we fine with using the action of the agent to reason acausally about the predictor? The algebra of partial Markov categories provides a mathematical framework where it makes sense to assume so; the algebra of message theories takes care of the rest of the discussion.

Bibliography. The idea of using send/receive effects for encoding sessions is not new. Message passing can be also seen as a core component of game semantics, which has a vast literature [AJ94, AM99, Hyl97]. Game semantics has the ambition to provide the mathematical structures that describe coordination between distributed agents, starting from a duality between the Player's moves and the Opponent's moves; one of its achievements is to provide syntax-independent semantics for different extensions of PCF [McC00], including one with global state [AHM98].

Game semantics and session types [HYC08] have been called "two sides of the same coin" [CY19]. Orchard and Yoshida [OY16] discuss two mutual embeddings between an effectful λ-calculus (PCF) and a session π-calculus; further work also implemented the corresponding do-notation [OY17]. In our framework, this correspondence occurs between premonoidal categories and message theories; and thus between do-notation and polar shuffles.

Particularly relevant is Melliès' categorical approach to game semantics in the form of *template games* [Mel19]. The crucial difference between the present proposal and Melliès' line of work is that it starts from labelled transition systems as the basic notion. Melliès introduces *asynchronous graphs* – graphs with a set of commuting squares – and many of the same ingredients that we use here. Asynchronous graphs explain shuffles, polarization, and a failure of interchange in the form of a Grey tensor product [Mel21]. We would be interested to compare our approach to message passing with the framework of template games, and especially the points where they diverge: we land on normal duoidal categories while template games are based on Girard's linear logic [Gir89].

Finally, Newcomb's problem (or *paradox*) was first stated by Nozick [Noz69]. Evidential decision theory is defended in the work of Ahmad [Ahm14]; both Everitt, Leike and Hutter [ELH15], and Yudkowsky and Soares [YS17] have formalized comparisons of evidential decision theory, causal decision theory, and further variants.

CHAPTER 5

Conclusions and Further Work

Conclusions

Monoidal Context Theory. We have universally characterized the normal produoidal category of monoidal lenses (Theorem 3.5.3) as a free normalization of the cofree produoidal category on top of a monoidal category. The interpretation of this result is relevant: the splice-contour adjunction ([MZ22], Theorem 2.4.4) relates each category to its cofree promonoidal category of incomplete terms; in the same way, we have constructed a monoidal splice-contour adjunction relating each monoidal category to its cofree produoidal category of incomplete processes. The category underlying this universal produoidal category is familiar: it is the category of monoidal lenses.

Monoidal lenses have gained recent popularity in applications of category theory, apart from their classical counterparts in database theory [JRW12]: they have spawned applications in bidirectional transformations [FGM+07] but also in functional programming [PGW17, CEG+20], open games [GHWZ18], polynomial functors [NS22] and quantum combs [HC22]. Moreover, a different promonoidal structure for lenses had been already studied in the past by Pastro and Street [PS07]. Apart from lenses, incomplete processes have appeared implicitly multiple times in recent literature. Kissinger and Uijlen [KU17] describe higher-order quantum processes using contexts with holes in compact closed monoidal categories. Ghani, Hedges, Winschel and Zahn [GHWZ18] describe economic game theory in terms of *lenses* and incomplete processes in cartesian monoidal categories. Bonchi, Piedeleu, Sobociński and Zanasi [BPSZ19] study contextual equivalence in their monoidal category of affine signal flow graphs. Di Lavore, de Felice and Román [DLdFR22] define *monoidal streams* by iterating monoidal context coalgebraically. This situation prompted a question: why are lenses so prevalent? why do they appear in seemingly unrelated applications? We can now claim a conceptual answer with a mathematical justification. Lenses are the universal algebra for decomposing morphisms in process theories. The recent applications of lenses all describe incomplete processes.

Incomplete processes have two uses: on the one hand, they track the dependencies between monoidal processes; on the other hand, they allow us to split the multiple agents of a multi-party process. The former is quite useful in itself: duoidal categories provide two different tensors – a sequential, ($\triangleleft$), and a parallel one ($\otimes$) – that can track dependencies between different processes (as we saw in Section 3.1.2). Signalling and non-signalling conditions are important to the study of quantum theories, and recent work by Wilson and Chiribella [WC22] and Hefford and Kissinger [HK22] has studied signalling using structures close to monoidal lenses; we hope that the universal characterization of lenses as cofree produoidal categories may help extracting the exact structure needed for these physical frameworks. Even forgetting about dependencies explicitly, the theory of monoidal contexts allows us to pursue branches of computer science that were classically restricted to 1-dimensional syntaxes: the recent work of Eanrshaw and Sobocinski [ES22] studies

monoidal regular languages as the natural monoidal analogue of the classical notion of regular language.

However, it is the second application of incomplete processes the one that this author found more surprising: we can now separate the different agents of a multi-party process in an arbitrary monoidal category. Message passing was not the main goal of this thesis, but developing it has brought interesting connections to *game semantics*.

Monoidal Message Passing. Symmetric monoidal categories have two operations: sequential composition ($\fatsemi$) and parallel composition ($\otimes$). Naively, we would think then that there are two ways of decomposing monoidal processes: sequentially and in parallel. This is not false, and for many applications this may be the simpler way of dealing with this; after all, glueing sequentially and in parallel is the only thing we need to separate a string diagram in its atomic parts. However, this misses the rich algebra of incomplete morphisms and their compositions: monoidal lenses can be composed according to any polar shuffle, and that is the algebra that we are implicitly using when we cut a string diagram into pieces that do not necessarily follow the sequential and parallel divisions. We can now argue that the algebra of message passing for process theories is that of message theories. Message theories try to be a minimalistic axiomatization of what it means to communicate different processes that send and receive messages: we have argued for these axioms in Section 4.1, in a way that should appeal any reader not familiar with the categorical framework behind them. Their combinatorial characterization in terms of polar shuffles only makes them more concrete.

We have developed a theory of context on top of monoidal categories and we have used it to develop a canonical theory of message passing in monoidal categories. We have argued that this a fundamental structure for concurrency, and we have characterized it universally; however, we could still argue that it does not address the problem posed by Abramsky of finding the fundamental structures of concurrency [Abr05]. The main limitation of this framework is that it does not provide most of the features that we expect from fully-fledged session types: how can we model choice, or synchrony [Hon93, HYC08]? how can we model iteration, feedback, or other common programming constructs [McC00]? Minimalism, however, may be a good thing to separate these concerns from the fundamental structure: if we want to model choice, we can do so using *distributive categories* [CLW93], or *linear actegories* [CP09]; if we want feedback and iteration, we can recall traced categories [JSV96], categories with feedback [LGR+23], and notions of monoidal automata [DLdFR22]. Precisely because our framework is minimal, it seems that it is robust enough to support these additions; we will be interested in constructing models of game semantics in further work.

Future Work. Monoidal game semantics was not the original goal of this project but it became its most promising avenue; last section gives us a recipe to construct a *session do-notation* calculus on top of any monoidal category, which would include stochastic, non-deterministic and partial variants of a multi-party do-notation calculus. Levy, Power and Thielecke [LPT03] discuss the correspondence between premonoidal categories and call-by-value languages; we would be interested in extending this correspondence into message passing [OY17]. Once there, it seems plausible that we can connect this idea to the literature on game semantics for fully-fledged programming semantics [McC00]. Specifically, we would like to construct a model of probabilistic programming allowing for message passing, choice and iteration; multiple developments in categorical probability (such as quasiborel spaces [HKS+18] or Markov categories [Fri20]) make this possible.

The existence of a vast literature on message passing in terms of linear actions [CP09] makes it particularly important to understand exactly how duoidal categories and linear logic relate. We conjecture that physical duoidal categories also form *isomix* linearly distributive categories [CS97a], and this may be a categorical justification behind this connection.

For conciseness, we have not discussed approaches to iteration in monoidal categories. Joint work of this author with Di Lavore and de Felice [DLdFR22] has shown that it is possible to reason coinductively with automata in monoidal categories, using precisely monoidal lenses to describe incomplete processes. It seems plausible then, that mixing coinduction with message passing allows us to talk about networks of stochastic iterative processes: these have remarkable applications in scientific modelling, where we can write causal networks of stochastic processes in which unknown components are approximated stochastically.

Bibliography

[Abr05] Samson Abramsky. What are the fundamental structures of concurrency?: We still don't know! In Luca Aceto and Andrew D. Gordon, editors, *Proceedings of the Workshop "Essays on Algebraic Process Calculi", APC 25, Bertinoro, Italy, August 1-5, 2005*, volume 162 of *Electronic Notes in Theoretical Computer Science*, pages 37–41. Elsevier, 2005. 11, 134

[AC09] Samson Abramsky and Bob Coecke. Categorical quantum mechanics. In Kurt Engesser, Dov M. Gabbay, and Daniel Lehmann, editors, *Handbook of Quantum Logic and Quantum Structures*, pages 261–323. Elsevier, Amsterdam, 2009. 18, 44

[AHLF18] Marcelo Aguiar, Mariana Haim, and Ignacio López Franco. Monads on higher monoidal categories. *Applied Categorical Structures*, 26(3):413–458, Jun 2018. 88

[AHM98] Samson Abramsky, Kohei Honda, and Guy McCusker. A fully abstract game semantics for general references. In *Proceedings. Thirteenth Annual IEEE Symposium on Logic in Computer Science (Cat. No. 98CB36226)*, pages 334–344. IEEE, 1998. 132

[Ahm14] Arif Ahmed. *Evidence, Decision and Causality*. Cambridge University Press, 2014. 129, 132

[AHS02] Samson Abramsky, Esfandiar Haghverdi, and Philip J. Scott. Geometry of interaction and linear combinatory algebras. *Math. Struct. Comput. Sci.*, 12(5):625–665, 2002. 44

[AJ94] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *The Journal of Symbolic Logic*, 59(2):543–574, 1994. 115, 132

[Alu21] Paolo Aluffi. *Algebra: Chapter 0*, volume 104. American Mathematical Society, 2021. 18

[AM] Marcelo Aguiar and Swapneel A. Mahajan. personal communication. 78

[AM99] Samson Abramsky and Guy McCusker. Game Semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, NATO ASI Series, pages 1–55, Berlin, Heidelberg, 1999. Springer. 120, 132

[AM10] Marcelo Aguiar and Swapneel A. Mahajan. *Monoidal functors, species and Hopf algebras*, volume 29. American Mathematical Society Providence, RI, 2010. 11, 74, 76, 78, 82

[BCST96] Robert F. Blute, Robin B. Cockett, Robert A. G. Seely, and Todd H. Trimble. Natural deduction and coherence for weakly distributive categories. *Journal of Pure and Applied Algebra*, 113(3):229–296, 1996. 44

[BD98] John C. Baez and James Dolan. Higher-dimensional algebra III. n-categories and the algebra of opetopes. *Advances in Mathematics*, 135(2):145–206, 1998. 109

[BE14] John C. Baez and Jason Erbele. Categories in control. *arXiv preprint arXiv:1405.6881*, 2014. 37

[Bec23] Jorge Becerra. Strictification and non-strictification of monoidal categories. *arXiv preprint arXiv:2303.16740*, 2023. 24

[Bén67] Jean Bénabou. Introduction to bicategories. In *Reports of the Midwest Category Seminar*, pages 1–77, Berlin, Heidelberg, 1967. Springer Berlin Heidelberg. 27

[Bén68] Jean Bénabou. Structures algébriques dans les catégories. *Cahiers de topologie et géometrie différentielle*, 10(1):1–126, 1968. 36

[Bén00] Jean Bénabou. Distributors at work. *Lecture notes written by Thomas Streicher*, 11, 2000. 58, 59

[BG18] Guillaume Boisseau and Jeremy Gibbons. What you needa know about Yoneda: Profunctor optics and the Yoneda lemma (functional pearl). *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–27, 2018. 95

[BGK+16] Filippo Bonchi, Fabio Gadducci, Aleks Kissinger, Paweł Sobociński, and Fabio Zanasi. Rewriting modulo symmetric monoidal structure. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 710–719, 2016. 28, 29, 30, 36

[BGMS21] John C. Baez, Fabrizio Genovese, Jade Master, and Michael Shulman. Categories of nets. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021. 28

[BK22] Anne Broadbent and Martti Karvonen. Categorical composable cryptography. In Patricia Bouyer and Lutz Schröder, editors, *Foundations of Software Science and Computation Structures - 25th International Conference, FOSSACS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings*, volume 13242 of *Lecture Notes in Computer Science*, pages 161–183. Springer, 2022. 10, 97, 98

[BNR22] Guillaume Boisseau, Chad Nester, and Mario Román. Cornering optics. In *ACT 2022*, volume abs/2205.00842, 2022. 16, 115

[BPS17] Filippo Bonchi, Dusko Pavlovic, and Pawel Sobocinski. Functorial semantics for relational theories. *CoRR*, abs/1711.08699, 2017. 11

[BPSZ19] Filippo Bonchi, Robin Piedeleu, Pawel Sobocinski, and Fabio Zanasi. Graphical affine algebra. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–12. IEEE, 2019. 133

[BR23] Dylan Braithwaite and Mario Román. Collages of string diagrams. *arXiv preprint arXiv:2305.02675*, 2023. 16, 62

[BS10] John C. Baez and Mike Stay. Physics, topology, logic and computation: A Rosetta stone. In *New Structures for Physics*, pages 95–172. Springer Berlin Heidelberg, 2010. 37

[BS13] Thomas Booker and Ross Street. Tannaka duality and convolution for duoidal categories. *Theory and Applications of Categories*, 28(6):166–205, 2013. 83

[BSS18] Filippo Bonchi, Jens Seeber, and Pawel Sobocinski. Graphical conjunctive queries. In Dan R. Ghica and Achim Jung, editors, *27th EACSL Annual Conference on Computer Science Logic, CSL 2018, September 4-7, 2018, Birmingham, UK*, volume 119 of *LIPIcs*, pages 13:1–13:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 18, 42, 44

[BSZ14] Filippo Bonchi, Paweł Sobociński, and Fabio Zanasi. A categorical semantics of signal flow graphs. In *International Conference on Concurrency Theory*, pages 435–450. Springer, 2014. 29

[BZ20] Nicolas Blanco and Noam Zeilberger. Bifibrations of polycategories and classical linear logic. In Patricia Johann, editor, *Proceedings of the 36th Conference on the Mathematical Foundations of Programming Semantics, MFPS 2020, Online, October 1, 2020*, volume 352 of *Electronic Notes in Theoretical Computer Science*, pages 29–52. Elsevier, 2020. 157

[Cam19] Alexander Campbell. How strict is strictification? *Journal of Pure and Applied Algebra*, 223(7):2948–2976, 2019. 24, 26

[CEG+20] Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregiàn, Bartosz Milewski, Emily Pillmore, and Mario Román. Profunctor optics, a categorical update. *CoRR*, abs/2001.07488, 2020. 15, 60, 94, 95, 99, 133

[CFS16] Bob Coecke, Tobias Fritz, and Robert W. Spekkens. A mathematical theory of resources. *Inf. Comput.*, 250:59–86, 2016. 18, 22, 96

[CGG+22] Geoffrey S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. In *European Symposium on Programming*, pages 1–28. Springer, Cham, 2022. 95

[CJ19] Kenta Cho and Bart Jacobs. Disintegration and Bayesian Inversion via String Diagrams. *Mathematical Structures in Computer Science*, pages 1–34, March 2019. 18, 42, 43

[CLW93] Aurelio Carboni, Stephen Lack, and Robert F.C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84(2):145–158, 1993. 134

[CP09] Robin B. Cockett and Craig A. Pastro. The logic of message-passing. *Sci. Comput. Program.*, 74(8):498–533, 2009. 134, 135

[CS97a] Robin B. Cockett and Robert A. G. Seely. Proof theory for full intuitionistic linear logic, bilinear logic, and mix categories. *Theory and Applications of categories*, 3(5):85–131, 1997. 135, 152

[CS97b] Robin B. Cockett and Robert A. G. Seely. Weakly distributive categories. *Journal of Pure and Applied Algebra*, 114(2):133–173, 1997. 108, 157

[CS07] Robin B. Cockett and Robert A. G. Seely. Polarized category theory, modules, and game semantics. *Theory and Applications of Categories*, 18(2):4–101, 2007. 12, 115

[CS09] Geoffrey S. H. Cruttwell and Michael Shulman. A unified framework for generalized multicategories. *arXiv preprint arXiv:0907.2460*, 2009. 109

[CW87] Aurelio Carboni and Robert F. C. Walters. Cartesian bicategories I. *Journal of pure and applied algebra*, 49(1-2):11–32, 1987. 22

[CY19] Simon Castellan and Nobuko Yoshida. Two sides of the same coin: session types and game semantics: a synchronous side and an asynchronous side. *Proceedings of the ACM on Programming Languages*, 3(POPL):1–29, 2019. 132

[Day70] Brian Day. On closed categories of functors. In *Reports of the Midwest Category Seminar IV*, volume 137, pages 1–38, Berlin, Heidelberg, 1970. Springer Berlin Heidelberg. 86, 88, 93, 155, 157

[Dd09] Mariangiola Dezani-Ciancaglini and Ugo de'Liguoro. Sessions and session types: An overview. In Cosimo Laneve and Jianwen Su, editors,

Web Services and Formal Methods, 6th International Workshop, WS-FM 2009, Bologna, Italy, September 4-5, 2009, Revised Selected Papers, volume 6194 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2009. 11

[DDR11] Jean-Guillaume Dumas, Dominique Duval, and Jean-Claude Reynaud. Cartesian effect categories are Freyd-categories. *Journal of Symbolic Computation*, 46(3):272–293, 2011. 47

[DGNO10] Vladimir Drinfeld, Shlomo Gelaki, Dmitri Nikshych, and Victor Ostrik. On braided fusion categories I. *Selecta Mathematica*, 16(1):1–119, 2010. 48

[DLdFR22] Elena Di Lavore, Giovanni de Felice, and Mario Román. Monoidal streams for dataflow programming. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. 133, 134, 135

[DLLNS21] Ivan Di Liberti, Fosco Loregian, Chad Nester, and Paweł Sobociński. Functorial semantics for partial theories. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–28, 2021. 11, 40

[DPS05] Brian Day, Elango Panchadcharam, and Ross Street. On centres and lax centres for promonoidal categories. In *Colloque International Charles Ehresmann*, volume 100, 2005. 83

[DS03] Brian Day and Ross Street. Quantum categories, star autonomy, and quantum groupoids, 2003. 155, 157

[EH61] Beno Eckman and Peter Hilton. Structure maps in group theory. *Fundamenta Mathematicae*, 50(2):207–221, 1961. 82

[EHR23] Matt Earnshaw, James Hefford, and Mario Román. The produoidal algebra of process decomposition. *arXiv preprint arXiv:2301.11867*, 2023. 16, 70, 83, 86, 88, 157

[ELH15] Tom Everitt, Jan Leike, and Marcus Hutter. Sequential extensions of causal and evidential decision theory. In *International Conference on Algorithmic Decision Theory*, pages 205–221. Springer, 2015. 132

[ES22] Matthew Earnshaw and Pawel Sobociński. Regular Monoidal Languages. In Stefan Szeider, Robert Ganian, and Alexandra Silva, editors, *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, volume 241 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:14, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 11, 88, 133

[FB94] David J Foulis and Mary K Bennett. Effect algebras and unsharp quantum logics. *Foundations of physics*, 24:1331–1352, 1994. 41, 43

[FGM+07] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bidirectional tree transformations: A linguistic approach to the view-update problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(3):17–es, 2007. 99, 133

[FGP21] Tobias Fritz, Tomáš Gonda, and Paolo Perrone. De Finetti's theorem in categorical probability. *Journal of Stochastic Analysis*, 2(4), 2021. 43

[FJ19] Brendan Fong and Michael Johnson. Lenses and learners. *arXiv preprint arXiv:1903.03671*, 2019. 95

[Fon13] Brendan Fong. Causal Theories: A Categorical Perspective on Bayesian Networks. *Master's Thesis, University of Oxford. ArXiv*

preprint arXiv:1301.6201, 2013. 43

[Fox76] Thomas Fox. Coalgebras and cartesian categories. *Communications in Algebra*, 4(7):665–667, 1976. 38, 39, 42

[FP19] Tobias Fritz and Paolo Perrone. A probability monad as the colimit of spaces of finite samples. *Theory and Applications of Categories*, 34(7):170–220, 2019. 43

[FPR21] Tobias Fritz, Paolo Perrone, and Sharwin Rezagholi. Probability, valuations, hyperspace: Three monads on top and the support as a morphism. *Mathematical Structures in Computer Science*, 31(8):850–897, 2021. 43

[FR20] Tobias Fritz and Eigil Fjeldgren Rischel. Infinite products and zero-one laws in categorical probability. *Compositionality*, 2:3, 2020. 43

[Fri20] Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020. 18, 43, 134

[FS19] Brendan Fong and David I Spivak. Supplying bells and whistles in symmetric monoidal categories. *arXiv preprint arXiv:1908.02633*, 2019. 39

[FV20] Ignacio López Franco and Christina Vasilakopoulou. Duoidal categories, measuring comonoids and enrichment. *arXiv preprint arXiv:2005.01340*, 2020. 77

[GF16] Richard Garner and Ignacio López Franco. Commutativity. *Journal of Pure and Applied Algebra*, 220(5):1707–1751, 2016. 11, 73, 76, 77, 78, 79, 82, 88, 92, 93

[GG09] Richard Garner and Nick Gurski. The low-dimensional structures formed by tricategories. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 146, pages 551–589. Cambridge University Press, 2009. 26

[GH99] Simon J. Gay and Malcolm Hole. Types and subtypes for client-server interactions. In S. Doaitse Swierstra, editor, *Programming Languages and Systems, 8th European Symposium on Programming, ESOP'99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, 22-28 March, 1999, Proceedings*, volume 1576 of *Lecture Notes in Computer Science*, pages 74–90. Springer, 1999. 108

[GHWZ18] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. Compositional game theory. In Anuj Dawar and Erich Grädel, editors, *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 472–481. ACM, 2018. 95, 99, 133

[Gir89] Jean-Yves Girard. Geometry of Interaction 1: Interpretation of System F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic Colloquium '88*, volume 127 of *Studies in Logic and the Foundations of Mathematics*, pages 221–260. Elsevier, 1989. 44, 132

[Gis88] Jay L. Gischer. The equational theory of pomsets. *Theoretical Computer Science*, 61(2-3):199–224, 1988. 82

[Gra81] Jan Grabowski. On partial languages. *Fundamenta Informaticae*, 4(2):427–498, 1981. 15, 81, 82

[Gro85] Alexandre Grothendieck. Récoltes et semailles: réflexions et témoignage sur un passé de mathématicien. *Grothendieck Circle Page*, 1985. 9

[Gui80] René Guitart. Tenseurs et machines. *Cahiers de topologie et géométrie différentielle catégoriques*, 21(1):5–62, 1980. 45

[Had18] Amar Hadzihasanovic. Weak units, universal cells, and coherence via universality for bicategories. *arXiv preprint arXiv:1803.06086*, 2018. 24

[Has97] Masahito Hasegawa. *Models of sharing graphs: a categorical semantics of let and letrec*. PhD thesis, University of Edinburgh, UK, 1997. 108

[HC22] James Hefford and Cole Comfort. Coend optics for quantum combs. *arXiv preprint arXiv:2205.09027*, 2022. 99, 133

[Hed19] Jules Hedges. Folklore: Monoidal kleisli categories, Apr 2019. 45

[Her00] Claudio Hermida. Representable multicategories. *Advances in Mathematics*, 151(2):164–225, 2000. 68

[Her01] Claudio Hermida. From coherent structures to universal properties. *Journal of Pure and Applied Algebra*, 165(1):7–61, 2001. 24

[HJ06] Chris Heunen and Bart Jacobs. Arrows, like monads, are monoids. In Stephen D. Brookes and Michael W. Mislove, editors, *Proceedings of the 22nd Annual Conference on Mathematical Foundations of Programming Semantics, MFPS 2006, Genova, Italy, May 23-27, 2006*, volume 158 of *Electronic Notes in Theoretical Computer Science*, pages 219–236. Elsevier, 2006. 13, 37, 61

[HJW+92] Paul Hudak, Simon L. Peyton Jones, Philip Wadler, Brian Boutel, Jon Fairbairn, Joseph H. Fasel, María M. Guzmán, Kevin Hammond, John Hughes, Thomas Johnsson, Richard B. Kieburtz, Rishiyur S. Nikhil, Will Partain, and John Peterson. Report on the Programming Language Haskell, A Non-strict, Purely Functional Language. *ACM SIGPLAN Notices*, 27(5):1, 1992. 13, 37

[HK22] James Hefford and Aleks Kissinger. On the pre- and promonoidal structure of spacetime. *arXiv preprint arXiv.2206.09678*, 2022. 80, 133

[HKS+18] Chris Heunen, Ohad Kammar, Sam Staton, Sean Moss, Matthijs Vákár, Adam Ścibior, and Hongseok Yang. The semantic structure of quasi-borel spaces. In *PPS Workshop on Probabilistic Programming Semantics*, 2018. 134

[HMH14] Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In Thomas A. Henzinger and Dale Miller, editors, *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 52:1–52:10. ACM, 2014. 44

[Hon93] Kohei Honda. Types for dyadic interaction. In Eike Best, editor, *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, volume 715 of *Lecture Notes in Computer Science*, pages 509–523. Springer, 1993. 11, 107, 134

[HR23] James Hefford and Mario Román. Optics for premonoidal categories. *CoRR*, abs/2305.02906, 2023. 16

[HS23] Chris Heunen and Jesse Sigal. Duoidally enriched Freyd categories. In *International Conference on Relational and Algebraic Methods in Computer Science*, pages 241–257. Springer, 2023. 80

[Huf54] David A. Huffman. The synthesis of sequential switching circuits. *Journal of the Franklin Institute*, 257(3):161–190, 1954. 13

[Hug00] John Hughes. Generalising monads to arrows. *Science of Computer Programming*, 37(1-3):67–111, 2000. 13, 17, 28, 30, 37, 44, 61

[Hug12] Dominic Hughes. Simple free star-autonomous categories and full coherence. *Journal of Pure and Applied Algebra*, 216(11):2386–2410, 2012. 120

[HV19] Chris Heunen and Jamie Vicary. *Categories for Quantum Theory: An Introduction*. Oxford University Press, 2019. 18

[HYC08] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 273–284. ACM, 2008. 107, 132, 134

[Hyl97] Martin Hyland. Game semantics. *Semantics and logics of computation*, 14:131, 1997. 132

[Jac15] Bart Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. *Logical Methods in Computer Science*, 11, 2015. 41, 43

[Jef97a] Alan Jeffrey. Premonoidal categories and a graphical view of programs. *Preprint at ResearchGate*, 1997. 13, 54

[Jef97b] Alan Jeffrey. Premonoidal categories and flow graphs. *Electronical Notes in Theoretical Computer Science*, 10:51, 1997. 46

[JHH09] Bart Jacobs, Chris Heunen, and Ichiro Hasuo. Categorical semantics for arrows. *J. Funct. Program.*, 19(3-4):403–438, 2009. 44, 61

[JKZ21] Bart Jacobs, Aleks Kissinger, and Fabio Zanasi. Causal inference via string diagram surgery: A diagrammatic approach to interventions and counterfactuals. *Mathematical Structures in Computer Science*, 31(5):553–574, 2021. 43

[JRW12] Michael Johnson, Robert Rosebrugh, and Richard J. Wood. Lenses, fibrations and universal translations. *Mathematical structures in computer science*, 22(1):25–42, 2012. 99, 133

[JS91] André Joyal and Ross Street. The geometry of tensor calculus, I. *Advances in Mathematics*, 88(1):55–112, 1991. 20, 21, 22, 24, 36, 44

[JS93] André Joyal and Ross Street. Braided tensor categories. *Advances in Mathematics*, 102(1):20–78, 1993. 22

[JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119:447 – 468, 04 1996. 134

[JZ20] Bart Jacobs and Fabio Zanasi. The logical essentials of bayesian reasoning. *Foundations of Probabilistic Programming*, pages 295–331, 2020. 43

[KL80] Gregory Kelly and Miguel Laplaza. Coherence for compact closed categories. *Journal of pure and applied algebra*, 19:193–213, 1980. 78

[Kme12] Edward Kmett. lens library, version 4.16. *Hackage https://hackage. haskell. org/package/lens-4.16*, 2018, 2012. 95

[KPT96] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. In Hans-Juergen Boehm and Guy L. Steele Jr., editors, *Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996*, pages 358–371. ACM Press, 1996. 108

[KSW97] Piergiulio Katis, Nicoletta Sabadini, and Robert F. C. Walters. Bicategories of processes. *Journal of Pure and Applied Algebra*, 115(2):141–178, 1997. 22

[KU17] Aleks Kissinger and Sander Uijlen. A categorical semantics for causal structure. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017. 133

[Lam69] Joachim Lambek. Deductive systems and categories II: standard constructions and closed categories. *Category Theory, Homology Theory and their Applications I*, 1969. 11, 68

[Lam86] Joachim Lambek. Cartesian closed categories and typed λ-calculi. In Guy Cousineau, Pierre-Louis Curien, and Bernard Robinet, editors, *Combinators and Functional Programming Languages*, Lecture Notes in Computer Science, pages 136–175, Berlin, Heidelberg, 1986. Springer. 11

[Lau05] Aaron D. Lauda. Frobenius algebras and ambidextrous adjunctions. *arXiv preprint math/0502550*, 2005. 155

[Law63] F. William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences*, 50(5):869–872, 1963. 10

[Lei04] Tom Leinster. *Higher Operads, Higher Categories*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004. 82

[Lev22] Paul Blain Levy. Call-by-push-value. *ACM SIGLOG News*, 9(2):7–29, may 2022. 46

[Lew06] Geoffrey Lewis. Coherence for a closed functor. In *Coherence in categories*, pages 148–195. Springer, 2006. 78

[LGR+21] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Pawel Sobocinski. A canonical algebra of open transition systems. In Gwen Salaün and Anton Wijs, editors, *Formal Aspects of Component Software - 17th International Conference, FACS 2021, Virtual Event, October 28-29, 2021, Proceedings*, volume 13077 of *Lecture Notes in Computer Science*, pages 63–81. Springer, 2021. 16

[LGR+23] Elena Di Lavore, Alessandro Gianola, Mario Román, Nicoletta Sabadini, and Pawel Sobocinski. Span(Graph): a canonical feedback algebra of open transition systems. *Softw. Syst. Model.*, 22(2):495–520, 2023. 16, 134

[Lor21] Fosco Loregiàn. *(Co)end Calculus*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2021. 58, 59, 61, 62

[LPT03] Paul Blain Levy, John Power, and Hayo Thielecke. Modelling environments in call-by-value programming languages. *Information and computation*, 185(2):182–210, 2003. 134

[LR23] Elena Di Lavore and Mario Román. Evidential decision theory via partial markov categories. In *LICS*, pages 1–14, 2023. 16, 40, 41, 42, 43

[LS09] F. William Lawvere and Stephen H. Schanuel. *Conceptual mathematics: a first introduction to categories*. Cambridge University Press, 2009. 9

[Mac63] Saunders MacLane. Natural associativity and commutativity. *Rice Institute Pamphlet-Rice University Studies*, 49(4), 1963. 22, 24, 36

[Mac78] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer New York, 1978. 22

[Mar14] Daniel Marsden. Category theory using string diagrams. *arXiv preprint arXiv:1401.7220*, 2014. 26, 27

[McC00] Guy McCusker. Games and full abstraction for FPC. *Information and Computation*, 160(1-2):1–61, 2000. 132, 134

[Mel19] Paul-André Melliès. Template games and differential linear logic. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*, pages 1–13. IEEE, 2019. 132

[Mel21] Paul-André Melliès. Asynchronous template games and the gray tensor product of 2-categories. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2021. 12, 132

[ML71] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer Verlag, 1971. 18, 23, 36, 59, 62

[MM90] José Meseguer and Ugo Montanari. Petri nets are monoids. *Inf. Comput.*, 88(2):105–155, 1990. 28

[Mog91] Eugenio Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, 1991. 13, 44, 61

[MP21] Cary Malkiewich and Kate Ponto. Coherence for bicategories, lax functors, and shadows, 2021. 78

[MS14] Rasmus Ejlers Møgelberg and Sam Staton. Linear usage of state. *Log. Methods Comput. Sci.*, 10(1), 2014. 54

[MS18] Paul-André Melliès and Léo Stefanesco. A game semantics of concurrent separation logic. *Electronic Notes in Theoretical Computer Science*, 336:241–256, 2018. 120

[Mye16] David Jaz Myers. String diagrams for double categories and equipments, 2016. 115

[MZ22] Paul-André Melliès and Noam Zeilberger. Parsing as a Lifting Problem and the Chomsky-Schützenberger Representation Theorem. In *MFPS 2022-38th conference on Mathematical Foundations for Programming Semantics*, 2022. 11, 57, 69, 70, 88, 133

[Nes21] Chad Nester. The structure of concurrent process histories. In Ferruccio Damiani and Ornela Dardha, editors, *Coordination Models and Languages - 23rd IFIP WG 6.1 International Conference, COORDINATION 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14-18, 2021, Proceedings*, volume 12717 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2021. 12, 114, 115

[Noz69] Robert Nozick. Newcomb's Problem and Two Principles of Choice. In *Essays in honor of Carl G. Hempel*, pages 114–146. Springer, 1969. 129, 132

[NS22] Nelson Niu and David I. Spivak. Polynomial functors: A general theory of interaction. *In preparation*, 2022. 99, 133

[NV23] Chad Nester and Niels F. W. Voorneveld. Protocol choice and iteration for the free cornering. *CoRR*, abs/2305.16899, 2023. 115

[Ord20] Toby Ord. *The precipice: Existential risk and the future of humanity.* Hachette Books, 2020. 9

[OY16] Dominic Orchard and Nobuko Yoshida. Effects as sessions, sessions as effects. *ACM SIGPLAN Notices*, 51(1):568–581, 2016. 12, 132

[OY17] Dominic Orchard and Nobuko Yoshida. Session types with linearity in Haskell. In Simon Gay and António Ravara, editors, *Behavioural Types: from Theory to Tools*, River Publishers Series in Automation, Control and Robotics. River Publishers, 2017. 132, 134

[Pat01] Ross Paterson. A new notation for arrows. In Benjamin C. Pierce, editor, *Proceedings of the Sixth ACM SIGPLAN International Conference on Functional Programming (ICFP '01), Firenze (Florence), Italy, September 3-5, 2001*, pages 229–240. ACM, 2001. 30, 61

[Pat03] Ross Paterson. Arrows and computation. *The Fun of Programming*, pages 201–222, 2003. 37

[Pav13] Dusko Pavlovic. Monoidal computer I: basic computability by string diagrams. *Inf. Comput.*, 226:94–116, 2013. 44

[PC07] Jorge Picado and Maria Manuel Clementino. *An Interview with William F. Lawvere.* Online, `https://www.mat.uc.pt/~picado/lawvere/interview.pdf`., 2007. 9

[PGW17] Matthew Pickering, Jeremy Gibbons, and Nicolas Wu. Profunctor optics: Modular data accessors. *Art Sci. Eng. Program.*, 1(2):7, 2017. 94, 95, 99, 133

[Pow02] John Power. Premonoidal categories as categories with algebraic structure. *Theor. Comput. Sci.*, 278(1-2):303–321, 2002. 13, 44

[PR84] Roger Penrose and Wolfgang Rindler. *Spinors and Spacetime.* Cited by Aleks Kissinger at the Categories mailing list. Cambridge University Press, 1984. 10

[PR97] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Math. Struct. Comput. Sci.*, 7(5):453–468, 1997. 45

[PS93] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. In *Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS '93), Montreal, Canada, June 19-23, 1993*, pages 376–385. IEEE Computer Society, 1993. 108

[PS07] Craig Pastro and Ross Street. Doubles for Monoidal Categories. *arXiv preprint arXiv:0711.1859*, 2007. 88, 99, 133

[PSV21] Evan Patterson, David I. Spivak, and Dmitry Vagner. Wiring diagrams as normal forms for computing in symmetric monoidal categories. *Electronic Proceedings in Theoretical Computer Science*, page 49–64, Feb 2021. 99

[PT99] John Power and Hayo Thielecke. Closed Freyd- and kappa-categories. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, volume 1644 of *Lecture Notes in Computer Science*, pages 625–634. Springer, 1999. 44, 45

[Ril18] Mitchell Riley. Categories of Optics. *arXiv preprint arXiv:1809.00738*, 2018. 94, 95, 97, 99

[Rom20a] Mario Román. Comb Diagrams for Discrete-Time Feedback. *CoRR*, abs/2003.06214, 2020. 99

[Rom20b] Mario Román. Open diagrams via coend calculus. In David I. Spivak and Jamie Vicary, editors, *Proceedings of the 3rd Annual International Applied Category Theory Conference 2020, ACT 2020, Cambridge, USA, 6-10th July 2020*, volume 333 of *EPTCS*, pages 65–78, 2020. 15, 62, 94, 96, 99

[Rom22] Mario Román. Promonads and string diagrams for effectful categories. In Jade Master and Martha Lewis, editors, *Proceedings Fifth International Conference on Applied Category Theory, ACT 2022, Glasgow, United Kingdom, 18-22 July 2022*, volume 380 of *EPTCS*, pages 344–361, 2022. 15, 16, 47

[Sel10] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010. 36

[Shu16] Michael Shulman. Categorical logic from a categorical point of view. *Available on the web*, 2016. 19, 20, 22, 32, 37, 65, 109, 111

[Shu17] Michael Shulman. Duoidal category (nlab entry), section 2., 2017. https://ncatlab.org/nlab/show/duoidal+category, Last accessed on 2022-12-14. 88, 109, 111

[Shu18] Michael Shulman. The 2-Chu-Dialectica construction and the polycategory of multivariable adjunctions. *arXiv preprint arXiv:1806.06082*, 2018. 49, 153

[SL13] Sam Staton and Paul Blain Levy. Universal properties of impure programming languages. In Roberto Giacobazzi and Radhia Cousot, editors, *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, pages 179–192. ACM, 2013. 44, 45

[Sob10] Paweł Sobociński. Representations of Petri net interactions. In *International Conference on Concurrency Theory*, pages 554–568. Springer, 2010. 22

[Sob13] Pawel Sobociński. Graphical linear algebra. Online, personal blog, https://graphicallinearalgebra.net, 2013. 20, 22

[Spi13] David I. Spivak. The operad of wiring diagrams: formalizing a graphical language for databases, recursion, and plug-and-play circuits. *CoRR*, abs/1305.0297, 2013. 81, 99, 109

[SS22] Brandon T. Shapiro and David I. Spivak. Duoidal structures for compositional dependence. *arXiv preprint arXiv:2210.01962*, 2022. 11, 73, 74, 76, 79, 80, 81, 82

[SSV20] Patrick Schultz, David I. Spivak, and Christina Vasilakopoulou. Dynamical systems and sheaves. *Applied Categorical Structures*, 28(1):1–57, 2020. 99

[Str04] Ross Street. Frobenius monads and pseudomonoids. *Journal of mathematical physics*, 45(10):3930–3948, 2004. 157

[Str12] Ross Street. Monoidal categories in, and linking, geometry and algebra. *Bulletin of the Belgian Mathematical Society-Simon Stevin*, 19(5):769–820, 2012. 11, 77, 82, 86, 88

[SW01] Davide Sangiorgi and David Walker. *The Pi-Calculus - a theory of mobile processes*. Cambridge University Press, 2001. 107

[Sza75] Manfred E. Szabo. Polycategories. *Communications in Algebra*, 3(8):663–689, 1975. 152, 157

[Tod10] Todd Trimble. Coherence theorem for monoidal categories (nlab entry), section 3. discussion, 2010. https://ncatlab.org/nlab/show/coherence+theorem+for+monoidal+categories, Last accessed on 2022-05-10. 49

[UV08] Tarmo Uustalu and Varmo Vene. Comonadic notions of computation. In Jiří Adámek and Clemens Kupke, editors, *Proceedings of the Ninth Workshop on Coalgebraic Methods in Computer Science, CMCS 2008, Budapest, Hungary, April 4-6, 2008*, volume 203 of *Electronic Notes in Theoretical Computer Science*, pages 263–284. Elsevier, 2008. 44

[UVZ18] Tarmo Uustalu, Niccolò Veltri, and Noam Zeilberger. The sequent calculus of skew monoidal categories. *Electronic Notes in Theoretical Computer Science*, 341:345–370, 2018. 11

[VC22] André Videla and Matteo Capucci. Lenses for composable servers. *CoRR*, abs/2203.15633, 2022. 99

[vdW21] John van de Wetering. A categorical construction of the real unit interval. *arXiv preprint arXiv:2106.10094*, 2021. 41, 43

[vN58] John von Neumann. *The Computer and the Brain*. Yale University Press, quoted by David Darlymple in "A New Type of Mathematics" (2018), a Transcript from a talk at Montreal, 1958. 9

[vS] Marilyn vos Savant. Parade 16: Ask Marilyn (Archived). https://web.archive.org/web/20130121183432/http://marilynvossavant.com/game-show-problem/. Accessed: 2013-01-21. 129

[WC22] Matt Wilson and Giulio Chiribella. A mathematical framework for transformations of physical processes. *arXiv preprint arXiv:2204.04319*, 2022. 133

[YS17] Eliezer Yudkowsky and Nate Soares. Functional Decision Theory: a New Theory of Instrumental Rationality. *ArXiv preprint arXiv:1710.05060*, 2017. 129, 132

Appendix

0.1. Coherence diagrams for a duoidal category

$$
\begin{array}{ccc}
((A \lhd B) \otimes (C \lhd D)) \otimes (E \lhd F) & \xrightarrow{\alpha} & (A \lhd B) \otimes ((C \lhd D) \otimes (E \lhd F)) \\
\downarrow{\scriptstyle \psi_2 \otimes id} & & \downarrow{\scriptstyle id \otimes \psi_2} \\
((A \otimes C) \lhd (B \otimes D)) \otimes (E \lhd F) & & (A \lhd B) \otimes ((C \otimes E) \lhd (D \otimes F)) \\
\downarrow{\scriptstyle \psi_2} & & \downarrow{\scriptstyle \psi_2} \\
((A \otimes C) \otimes E) \lhd ((B \otimes D) \otimes F) & \xrightarrow{\alpha \lhd \alpha} & (A \otimes (C \otimes E)) \lhd (B \otimes (D \otimes F)) \\
((A \lhd B) \lhd C) \otimes ((D \lhd E) \lhd F) & \xrightarrow{\beta \otimes \beta} & (A \lhd (B \lhd C)) \otimes (D \lhd (E \lhd F)) \\
\downarrow{\scriptstyle \psi_2} & & \downarrow{\scriptstyle \psi_2} \\
((A \lhd B) \otimes (D \lhd E)) \lhd (C \otimes F) & & (A \otimes D) \lhd ((B \lhd C) \otimes (E \lhd F)) \\
\downarrow{\scriptstyle \psi_2 \otimes id} & & \downarrow{\scriptstyle id \otimes \psi_2} \\
((A \otimes D) \lhd (B \otimes E)) \lhd (C \otimes F) & \xrightarrow{\beta} & (A \otimes D) \lhd ((B \otimes E) \lhd (C \otimes F))
\end{array}
$$

FIGURE 1. Coherence diagrams for associativity of a duoidal category.

$$
\begin{array}{ccc}
I \otimes (A \lhd B) & \xrightarrow{\psi_0 \otimes id} & (I \lhd I) \otimes (A \lhd B) \\
\downarrow{\scriptstyle \lambda} & & \downarrow{\scriptstyle \psi_2} \\
A \lhd B & \xleftarrow[\lambda \lhd \lambda]{} & (I \otimes A) \lhd (I \otimes B)
\end{array}
\qquad
\begin{array}{ccc}
(A \lhd B) \otimes I & \xrightarrow{\psi_0 \otimes id} & (A \lhd B) \otimes (I \lhd I) \\
\downarrow{\scriptstyle \rho} & & \downarrow{\scriptstyle \psi_2} \\
A \lhd B & \xleftarrow[\rho \lhd \rho]{} & (A \otimes I) \lhd (B \otimes I)
\end{array}
$$

FIGURE 2. Coherence diagrams for $\otimes$-unitality of a duoidal category.

$$
\begin{array}{ccc}
N \lhd (A \otimes B) & \xleftarrow{\varphi_2 \lhd id} & (N \otimes N) \lhd (A \otimes B) \\
\downarrow{\scriptstyle \kappa} & & \downarrow{\scriptstyle \psi_2} \\
A \otimes B & \xleftarrow[\kappa \otimes \kappa]{} & (N \lhd A) \otimes (N \lhd B) \\
(A \otimes B) \lhd N & \xleftarrow{id \lhd \varphi_2} & (A \otimes B) \lhd (N \otimes N) \\
\downarrow{\scriptstyle \nu} & & \downarrow{\scriptstyle \psi_2} \\
A \otimes B & \xleftarrow[\nu \otimes \nu]{} & (A \lhd N) \otimes (B \lhd N)
\end{array}
$$

FIGURE 3. Coherence diagrams for $\lhd$-unitality of a duoidal category.

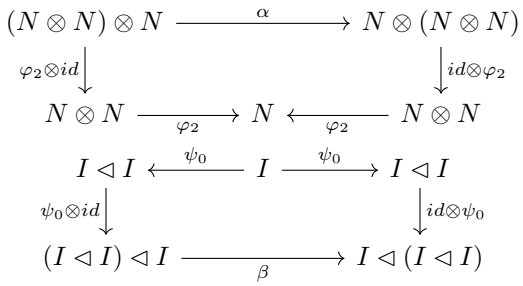


FIGURE 4. Associativity and coassociativity for N and I in a duoidal category.

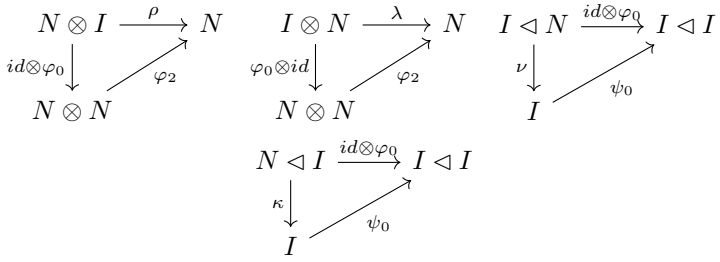


FIGURE 5. Unitality and counitality for N and I in a duoidal category.

0.2. Polycategories

This extra section repeats the splice-contour adjunction for polycategories. It is a detour from the main text; and it is not necessary for its development: this does not seem to be the direction we want to follow to study context in categories or monoidal categories. It is, however, another proof of the resilience of the splice-contour adjunction: the duality between a category and its opposite induces a pseudofrobenius algebra on the monoidal bicategory of profunctors.

0.2.1. Polycategories. A polycategory is like a category where every morphism has both a list of inputs and a list of outputs [Sza75]. This does not mean that its inputs and outpuss start forming a monoid, as in strict monoidal categories; morphisms really have different multiple inputs and outputs, and we need to choose a single one to compose along it. A polycategory, $\mathbb{P}$, contains a set of objects, $\mathbb{P}_{obj}$, as categories and multicategories; but instead of a set of morphisms, it will have a set of *polymorphisms*,

$$\mathbb{P}(X_1, \dots, X_n; Y_1, \dots, Y_m),$$

for each two lists of objects $X_1, \dots, X_n, Y_0, \dots, Y_m \in \mathbb{P}_{obj}$. As in linear logic, we denote both sides of a derivation by two metavariables, $\Gamma = X_1, \dots, X_n$ and $\Delta = Y_0, \dots, Y_m$, and write $\mathbb{P}(\Gamma; \Delta)$ for the set of polymorphisms.

Definition 0.2.1. A *polycategory* $\mathbb{P}$ is a collection of objects, $\mathbb{P}_{obj}$, together with a collection of *polymorphisms*, $\mathbb{P}(X_0, \dots, X_n; Y_0, \dots, Y_m)$, for each two lists of objects $X_0, \dots, X_n \in \mathbb{P}$ and $Y_0, \dots, Y_m \in \mathbb{P}$. For each object $X \in \mathbb{P}_{obj}$, there must be an identity, $\mathrm{id}_X \in \mathbb{P}_{obj}(X; X)$; and for each pair of composable maps,

$$f \in \mathbb{P}(\Gamma; \Delta_1, X, \Delta_2) \text{ and } g \in \mathbb{P}(\Gamma_1, X, \Gamma_2; \Delta),$$
$$\text{where either } \Delta_1 \text{ or } \Gamma_1, \text{ and either } \Delta_2 \text{ or } \Gamma_2, \text{ are empty,}$$

there must be a composite polymorphism $f \fatsemi_X g \in \mathbb{P}(\Gamma_1, \Gamma, \Gamma_2; \Delta_1, \Delta, \Delta_2)$. This means that there are four possible types of composition (Figure 6), and they yield the same polymorphism whenever they overlap.

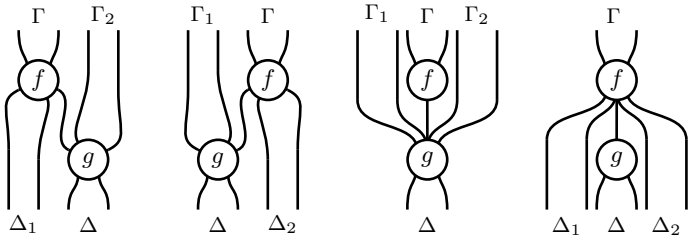


FIGURE 6. Four planar polycategorical compositions.

Moreover, polycategories must satisfy the following two unitality axioms, $f \fatsemi_X \mathrm{id}_X = f$ and $\mathrm{id}_X \fatsemi_X f = f$; two associativity axioms, $f \fatsemi_X (g \fatsemi_Y h) = (f \fatsemi_X g) \fatsemi_Y h$ and $f \fatsemi_X (g \fatsemi_Y h) = g \fatsemi_Y (f \fatsemi_X h)$; and an interchange axiom, $(f \fatsemi_A g) \fatsemi_B h = (f \fatsemi_B h) \fatsemi_A g$; whenever any of these is formally well-typed.

Remark 0.2.2. Asking for an identity on each object, $\mathrm{id}_X \in \mathbb{P}(X; X)$, is different from asking for an identity on each list of objects,

$$\mathrm{id}_{X_0, \dots, X_n} \in \mathbb{P}(X_0, \dots, X_n; X_0, \dots, X_n).$$

The latter gives rise to *isomix* categories [CS97a] and we will not discuss it here.

Example 0.2.3. A *polyfunctional relation*, $R \in \mathbf{MultiFun}(A_1, \dots, A_n; B_1, \dots, B_m)$, is a relation $R\colon A_1 \times \dots \times A_n \to B_1 \times \dots \times B_m$ together with representing functions that, given an element of the relation missing exactly one element, return the element missing. Explicitly, there exist two families of functions,

$$f_j\colon A_1 \times \dots \times A_n \times B_0 \times \cancel{B_j} \times \dots \times B_m \to B_j \quad \text{and}$$
$$g_i\colon A_1 \times \cancel{A_i} \times \dots \times A_n \times B_0 \times \dots \times B_m \to A_i,$$

such that $R(a_1, \dots, b_m)$ if and only if $f_j(a_1, \dots, b_m) = b_j$ and if and only if $g_i(a_1, \dots, b_m) = a_i$ for each two indices i and j. Polyfunctional relations form a polycategory with relational composition.

A $(1,1)$-polyfunctional relation is a pair of inverse functions. A (2,1) or (1,2)-polyfunctional relation is a triple functions $f_0\colon A_1 \times A_2 \to A_0$, $f_1\colon A_2 \times A_0 \to A_1$ and $f_2\colon A_0 \times A_1 \to A_2$ such that $f_1(a_1, a_2) = a_0$ if and only if $f_1(a_2, a_0) = a_1$ and if and only if $f_2(a_0, a_1) = a_2$. Polyfunctional relations are a decategorification of the *multivariable adjunctions* in the work of Shulman [Shu18].

0.2.2. The Category of Polycategories. Analogously to the categorical and multicategorical case, the theory of polyfunctors and polynatural transformations is synthetised in the 2-category **PolyCat** of polycategories, polyfunctors and polynatural transformations.

Definition 0.2.4. A *polyfunctor*, $F\colon \mathbb{P} \to \mathbb{Q}$, between two polycategories $\mathbb{P}$ and $\mathbb{Q}$, is an assignment on objects, $F_{obj}\colon \mathbb{P}_{obj} \to \mathbb{Q}_{obj}$ together with an assignment on polymorphisms

$$F_{n,m}\colon \mathbb{P}(X_0, \dots, X_n; Y_0, \dots, Y_m) \to \mathbb{Q}(F_{obj}X_0, \dots, F_{obj}X_n; F_{obj}Y_0, \dots, F_{obj}Y_m).$$

This assignment must be functorial, in that $F(f \fatsemi_{X_i} g) = F(f) \fatsemi_{X_i} F(g)$ and that $F(\mathrm{id}_X) = \mathrm{id}_{FX}$, whenever these are formally well-typed.

Definition 0.2.5. A polynatural transformation $\theta\colon F \to G$ between two polyfunctors $F, G\colon \mathbb{P} \to \mathbb{Q}$ is given by a family of polymorphisms $\theta_X \in \mathbb{Q}(FX; GX)$ such that, for each polymorphism $f \in \mathbb{P}(X_1, \dots, X_n; Y_1, \dots, Y_m)$, the following naturality condition holds

$$\theta_{X_1} \fatsemi \dots \fatsemi \theta_{X_n} \fatsemi G(f) = F(f) \fatsemi \theta_{Y_1} \fatsemi \dots \fatsemi \theta_{Y_m}.$$

Definition 0.2.6. Polyfunctors between polycategories form a category, **PolyCat**. This is moreover a 2-category with polynatural transformations.

Remark 0.2.7. In the same sense that a multifunctor from the terminal multicategory picks a monoid, a polyfunctor from the terminal polycategory should pick a *polyoid* – instead, we call these *Frobenius monoids*.

0.2.3. Polycategorical Contour.

Definition 0.2.8. Let $\mathbb{P}$ be a polycategory. Its contour, $\mathsf{Contour}\mathbb{P}$, is the category presented by the following generators and equations:

- two polarized objects, X^ℓ and X^r, for each object $X \in \mathbb{P}_{obj}$;
- for each polymorphism, $f \in \mathbb{P}(X_1, \dots, X_n; Y_1, \dots, Y_m)$, the following generators,

$$f^r_1\colon X^r_1 \to X^\ell_2, \quad \dots, \quad f^r_{n-1}\colon X^r_{n-1} \to X^\ell_n, \quad f^d\colon X^r_n \to Y^r_m,$$
$$f^\ell_1\colon Y^\ell_2 \to Y^r_1, \quad \dots, \quad f^\ell_{m-1}\colon Y^\ell_m \to Y^r_{m-1}, \quad f^u\colon Y^\ell_1 \to X^\ell_1,$$

  having instead $f^u\colon Y^\ell_1 \to Y^r_m$ when $n = 0$, having instead $f^d\colon X^r_n \to X^\ell_1$ when $m = 0$, and using no generators for $(0,0)$-polymorphisms;

to which we impose equations requiring contour to preserve identities, $(\mathrm{id}_X)^u = \mathrm{id}_{X^\ell}$ and $(\mathrm{id}_X)^d = \mathrm{id}_{X^r}$; and requiring contour to preserve compositions, meaning that for each $f \in \mathbb{P}(X_1, \dots, X_n; Y_1, \dots, Y_m)$ and each $g \in \mathbb{P}(Z_1, \dots, Z_p; Q_1, \dots, Q_q)$ such that $Y_m = U_1$, the contour of the composition along $Y_m = U_1$ is defined by the following eight cases

$$
\begin{aligned}
&(f \mathbin{⅋_{X_i}} g)^u = f^u; && (f \mathbin{⅋_{X_i}} g)^r_i = f^r_i, \text{ for } i = 1, \dots, n-1;\\
&(f \mathbin{⅋_{Y_m}} g)^r_n = f^d \mathbin{⅋} g^r_1; && (f \mathbin{⅋_{Y_m}} g)^r_j = g^r_{j-n+1}, \text{ for } j = n+1, \dots, n+p-2;\\
&(f \mathbin{⅋_{Y_m}} g)^d = g^d; && (f \mathbin{⅋_{Y_m}} g)^\ell_i = f^\ell_i, \text{ for } i = 1, \dots, m-2;\\
&(f \mathbin{⅋_{Y_m}} g)^d_{m-1} = g^u \mathbin{⅋} f^\ell_{m-1}; && (f \mathbin{⅋_{Y_m}} g)^d_j = g^\ell_{j-m+1}, \text{ for } j = m+1, \dots, m+q-2;
\end{aligned}
$$

and similar conditions for the rest of the compositions. These equations are depicted in Figure 7.

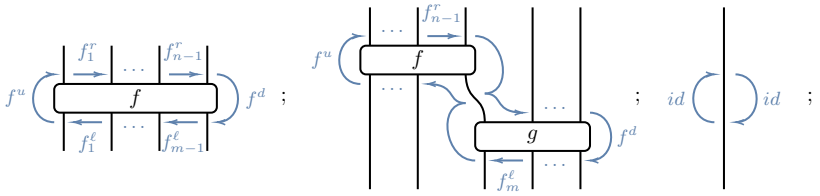


FIGURE 7. Contour of a morphism, composition of contours, and identity contours.

Proposition 0.2.9. *Contouring extends to a functor from the category of polycategories to the category of categories,* $\mathcal{C}\colon$ **PolyCat** $\to$ **Cat**.

0.2.4. Malleable Polycategories. A malleable polycategory is a polycategory where each morphism can be morphed uniquely into any possible shape. This means that there exist unique factorizations of each morphism into each one of the possible shapes for composition.

Definition 0.2.10. The (1,1)-polymorphisms of a polycategory $\mathbb{P}$ form an underlying category $\mathbb{P}^u$. The polymorphisms form profunctors over the polycategory and their composition, in its four possible forms, is dinatural with respect to the underlying category. This means that the following four operations are well-defined:

$$
\begin{aligned}
(⅋)_1 &\colon \left(\int^{X \in \mathbb{P}^u} \mathbb{P}(\Gamma; \Delta, X) \times \mathbb{P}(X, \Gamma'; \Delta')\right) \to \mathbb{P}(\Gamma, \Gamma'; \Delta, \Delta'),\\
(⅋)_2 &\colon \left(\int^{X \in \mathbb{P}^u} \mathbb{P}(\Gamma; X, \Delta) \times \mathbb{P}(\Gamma', X; \Delta')\right) \to \mathbb{P}(\Gamma, \Gamma'; \Delta, \Delta'),\\
(⅋)_3 &\colon \left(\int^{X \in \mathbb{P}^u} \mathbb{P}(\Gamma; \Delta_1, X, \Delta_2) \times \mathbb{P}(X; \Delta)\right) \to \mathbb{P}(\Gamma; \Delta_1, \Delta, \Delta_2),\\
(⅋)_4 &\colon \left(\int^{X \in \mathbb{P}^u} \mathbb{P}(\Gamma; X) \times \mathbb{P}(\Gamma_1, X, \Gamma_2; \Delta)\right) \to \mathbb{P}(\Gamma_1, \Gamma, \Gamma_2; \Delta).
\end{aligned}
$$

Definition 0.2.11. A *malleable polycategory* is a polycategory where dinatural composition, in all its four forms, is invertible.

Remark 0.2.12. If a polycategory is malleable, we can reconstruct it up to isomorphism from its binary, cobinary, nullary and conullary maps. When defining a malleable polycategory, it is usually easier to provide these binary, cobinary, unary, nullary and conullary maps, and deduce from those the rest of the structure. The situation is now similar to that of linearly distributive categories: we do not need to provide all n-ary tensors in order to define a linearly distributive category, we only provide the binary $(\otimes, ⅋)$ and unary (I, Z) tensors.

This suggests that we will really work with a *biased* version of malleable polycategories, one that privileges the binary and nullary tensors over the others. Biased malleable polycategories are what we will call *prostar autonomous categories*.

0.2.5. Prostar-Autonomous Categories. Prostar-autonomous categories provide an algebra for both coherent composition and decomposition. Apart from the usual *morphisms*, $\mathbb{V}(X;Y)$; and the *joints*, $\mathbb{V}(X_0 \otimes X_1;Y)$, and *units*, $\mathbb{V}(\top;Y)$, of a promonoidal category; a prostar-autonomous category has *splits*, $\mathbb{V}(X;Y_0 \parr Y_1)$, and *atoms*, $\mathbb{V}(X;\bot)$. As in the case of multicategories, these compositions and decompositions must be coherent, which translates into the existence of natural isomorphisms witnessing a Frobenius rule, the *Frobenius distributors*

$$\varphi_l \colon \int^W \mathbb{C}(A;C \parr W) \times \mathbb{C}(W \otimes B;D) \xrightarrow{\cong} \mathbb{C}(A \otimes B;C \parr D), \text{ and}$$
$$\varphi_r \colon \int^W \mathbb{C}(A \otimes W;C) \times \mathbb{C}(B;W \parr D) \xrightarrow{\cong} \mathbb{C}(A \otimes B;C \parr D).$$

In summary, after this section, we will have developed the relation between malleability and profunctorial structures in an analogous way for both multicategories and polycategories.

Multicategory	Malleable Multicategory	Promonoidal category
Polycategory	Malleable Polycategory	Prostar autonomous category

Definition 0.2.13. Prostar-autonomous categories are the 2-Frobenius monoids of the monoidal bicategory of profunctors, which is equivalent to the following definition. A *prostar autonomous category* is a category $\mathbb{C}$ endowed with a promonoidal structure $(\mathbb{C}, \otimes, \top)$, and procomonoidal structure $(\mathbb{C}, \parr, \bot)$, that interact as a Frobenius pseudomonoid [DS03, Lau05]. That is, it is a category endowed with four profunctors, suggestively written $\mathbb{C}(\bullet \otimes \bullet;\bullet)$, $\mathbb{C}(\top;\bullet)$, $\mathbb{C}(\bullet;\bot)$ and $\mathbb{C}(\bullet;\bullet \parr \bullet)$, as if they were representable. These profunctors form two promonoidal categories [Day70] with coherent associators and unitors. Further, they are endowed with invertible Frobenius distributors,

$$\varphi_l \colon \int^W \mathbb{C}(A;C \parr W) \times \mathbb{C}(W \otimes B;D) \xrightarrow{\cong} \mathbb{C}(A \otimes B;C \parr D),$$
$$\varphi_r \colon \int^W \mathbb{C}(A \otimes W;C) \times \mathbb{C}(B;W \parr D) \xrightarrow{\cong} \mathbb{C}(A \otimes B;C \parr D),$$

such that every formal diagram formed of these distributors and promonoidal coherences commutes.

Prostar autonomous categories have a canonical *prostar* given by profunctors $\mathbb{C}(\bullet \otimes \bullet;\bot)$ and $\mathbb{C}(\top;\bullet \parr \bullet)$. We may think of a prostar autonomous category as a category $\mathbb{C}$ equipped with sets of polymorphisms $\mathbb{C}(\bullet \otimes \ldots \otimes \bullet;\bullet \parr \ldots \parr \bullet)$. The Frobenius isomorphisms let us decompose polymorphisms into combinations of the pro(co)monoidal structures: this decomposition is unique up to dinaturality. Informally, prostar autonomous categories are to polycategories what promonoidal categories are to (co)multicategories.

Definition 0.2.14. A *prostar functor* $F \colon \mathbb{V} \to \mathbb{W}$ is a quintuple $(F_{obj}, F_\otimes, F_\parr, F_\top, F_\bot)$ where $(F_{obj}, F_\otimes, F_\top)$ and $(F_{obj}, F_\parr, F_\bot)$ are promonoidal functors that together strictly preserve the Frobenius distributors, in that $\varphi_l \fatsemi (F_\otimes \times F_\parr) = (F_\parr \times F_\otimes) \fatsemi \varphi'_l$ and $\varphi_r \fatsemi (F_\otimes \times F_\parr) = (F_\otimes \times F_\parr) \fatsemi \varphi'_r$. Prostar functors between prostar autonomous categories form a category, **ProStar**.

0.2.6. Prostar Autonomous are Malleable Polycategories. In this section, we show that the category of prostar autonomous categories is equivalent to that of malleable polycategories. In this sense, the study of malleable polycategories is the study of prostar autonomous categories.

Definition 0.2.15 (Polycategorical analogue of Definition 2.3.9)**.** Let $\mathbb{W}$ be a prostar autonomous category. There is a malleable polycategory, $\mathbb{W}^m$, that has the same objects but polymorphisms defined by the elements of the prostar autonomous category. By induction, we define

$$\begin{aligned}
\mathbb{W}^m(X_0, X_1, \Gamma; \Delta) &= \int^V \mathbb{W}(X_0 \otimes X_1; V) \times \mathbb{W}^m(V, \Gamma; \Delta),\\
\mathbb{W}^m(; \Delta) &= \int^V \mathbb{W}(\top; V) \times \mathbb{W}(V; \Delta),\\
\mathbb{W}^m(X; Y_0, Y_1, \Delta) &= \int^V \mathbb{W}^m(X; V, \Delta) \times \mathbb{W}(V; Y_0 \parr Y_1),\\
\mathbb{W}^m(X;) &= \mathbb{W}(X; \bot).
\end{aligned}$$

In other words, the polymorphisms are elements of the left-biased tree reductions of the promonoidal category, seen as a 2-monoid. The four forms of dinatural composition are then defined to be the unique map relating two tree expressions in a 2-Frobenius monoid, which exist uniquely by coherence,

$$\begin{aligned}
(\mathrm{coh})_1 &\colon \left(\int^{X \in \mathbb{W}^m} \mathbb{W}^m(\Gamma; \Delta, X) \times \mathbb{W}^m(X, \Gamma'; \Delta')\right) \to \mathbb{W}^m(\Gamma, \Gamma'; \Delta, \Delta'),\\
(\mathrm{coh})_2 &\colon \left(\int^{X \in \mathbb{W}^m} \mathbb{W}^m(\Gamma; X, \Delta) \times \mathbb{W}^m(\Gamma', X; \Delta')\right) \to \mathbb{W}^m(\Gamma, \Gamma'; \Delta, \Delta'),\\
(\mathrm{coh})_3 &\colon \left(\int^{X \in \mathbb{W}^m} \mathbb{W}^m(\Gamma; \Delta_1, X, \Delta_2) \times \mathbb{W}^m(X; \Delta)\right) \to \mathbb{W}^m(\Gamma; \Delta_1, \Delta, \Delta_2),\\
(\mathrm{coh})_4 &\colon \left(\int^{X \in \mathbb{W}^m} \mathbb{W}^m(\Gamma; X) \times \mathbb{W}^m(\Gamma_1, X, \Gamma_2; \Delta)\right) \to \mathbb{W}^m(\Gamma_1, \Gamma, \Gamma_2; \Delta).
\end{aligned}$$

Coherence maps are isomorphisms, and so dinatural composition is invertible, making the polycategory malleable. By coherence for pseudomonoids, composition must satisfy associativity and unitality.

Proposition 0.2.16. *The category of prostar autonomous categories and the category of malleable polycategories are equivalent with the functor* $(\bullet)^m \colon$ **ProStar** $\to$ **mPoly** *induced by the construction of the underlying malleable polycategory of a prostar autonomous category. This is the polycategorical analogue of Proposition 2.3.10.*

Proof. First, let us show that a prostar functor, $F \colon \mathbb{V} \to \mathbb{W}$, induces a polyfunctor, $F^m \colon \mathbb{V}^m \to \mathbb{W}^m$, between the Underlying polycategories. On objects, we define it to be the same, $F^m_{obj} = F_{obj}$. On polymorphisms, we can define the binary, nullary, cobinary, conullary and unary using the prostar functor structure,

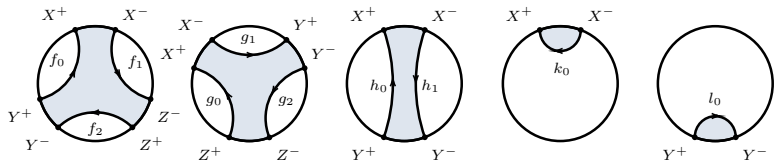
$$F^m_{2,1} = F_\otimes;\ F^m_{0,1} = F_\top;\ F^m_{1,2} = F_\parr;\ F^m_{1,0} = F_\bot;\ \text{and } F^m_{1,1} = F.$$

□

0.2.7. Splice of a Polycategory.

Definition 0.2.17. Let $\mathbb{C}$ be a category. Its prostar autonomous category of *spliced arrows*, $S\mathbb{C}$, has underlying category $\mathbb{C}^{\mathrm{op}} \times \mathbb{C}$. Intuitively, its profunctors are defined by spliced circles of morphisms.

Explicitly, it is defined by the following profunctors (below, left). The coherence isomorphisms are defined by glueing circles along the desired boundary and composing the relevant arrows; two compositions are isomorphic if and only if they determine the same arrows (below, right).



Remark 0.2.18. This structure appeared in Day & Street [DS03, Ex. 7.3], where it was noticed that the canonical promonoidal category induced by a small category [Day70] has an involution. As a multicategory, it was rediscovered by Melliès & Zeilberger [**?**]. Monoidal spliced arrows were explicitly introduced and characterized as an adjunction in a joint work [EHR23].

$$\begin{aligned}
\mathsf{Splice}(\mathbb{C})\left(\begin{smallmatrix} X^+ \\ X^- \end{smallmatrix}; \begin{smallmatrix} Y^+ \\ Y^- \end{smallmatrix} \parr \begin{smallmatrix} Z^+ \\ Z^- \end{smallmatrix}\right) &= \mathbb{C}(Y^+; X^+) \times \mathbb{C}(X^-; Z^-) \times \mathbb{C}(Z^+; Y^-); \\
\mathsf{Splice}(\mathbb{C})\left(\begin{smallmatrix} X^+ \\ X^- \end{smallmatrix} \otimes \begin{smallmatrix} Y^+ \\ Y^- \end{smallmatrix}; \begin{smallmatrix} Z^+ \\ Z^- \end{smallmatrix}\right) &= \mathbb{C}(Z^+; X^+) \times \mathbb{C}(X^-; Y^+) \times \mathbb{C}(Y^-; Z^-); \\
\mathsf{Splice}(\mathbb{C})\left(\begin{smallmatrix} X^+ \\ X^- \end{smallmatrix}; \begin{smallmatrix} Y^+ \\ Y^- \end{smallmatrix}\right) &= \mathbb{C}(Y^+; X^+) \times \mathbb{C}(X^-; Y^-); \\
\mathsf{Splice}(\mathbb{C})\left(\begin{smallmatrix} X^+ \\ X^- \end{smallmatrix}; \bot\right) &= \mathbb{C}(X^-; X^+); \\
\mathsf{Splice}(\mathbb{C})\left(\top; \begin{smallmatrix} Y^+ \\ Y^- \end{smallmatrix}\right) &= \mathbb{C}(Y^+; Y^-).
\end{aligned}$$

Remark 0.2.19. $\mathsf{Splice}(\mathbb{C})$ has a representable prostar, given on objects by

$$\begin{pmatrix} X^+ \\ X^- \end{pmatrix}^* = \begin{pmatrix} X^- \\ X^+ \end{pmatrix}.$$

Proposition 0.2.20. *Spliced arrows extend to a functor,* $\mathsf{Splice} : \mathbf{Cat} \to \mathbf{ProStar}$.

Theorem 0.2.21. *Contour extends to a functor* $\mathsf{Contour}\colon \mathbf{PolyCat} \to \mathbf{Cat}$*, splice extends to a functor* $\mathsf{Splice}\colon \mathbf{Cat} \to \mathbf{ProStar}$. $\mathsf{Contour}$ *is left adjoint to* Splice *composed with the forgetful functor,* $\mathsf{Contour} \dashv \mathsf{Splice} \fatsemi \mathsf{Forget}$*; and* $\mathsf{Contour}$ *composed with the forgetful functor is left adjoint to* Splice*, meaning* $\mathsf{Forget} \fatsemi \mathsf{Contour} \dashv \mathsf{Splice}$.

Proof. The proof extends our previous one [EHR23, Theorem 3.7]. □

0.2.8. Bibliography. Polycategories were defined by Szabo [Sza75] in the symmetric case; Cockett and Seely contributed the planar version we study here [CS97b, BZ20].

Street [Str04] prove that Frobenius pseudomonoids in **Prof** are equivalent to what Day & Street [DS03] call "$*$-autonomous promonoidal categories". The minor twist we take, "prostar autonomous", emphasizes that the canonical prostar may not

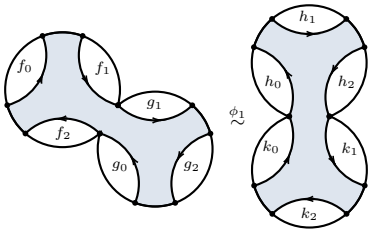


Figure 8. Holds if $f_0 = k_0 \fatsemi h_0$, $f_1 \fatsemi g_1 = h_1$, $g_2 = h_2 \fatsemi k_1$, $g_0 \fatsemi f_2 = k_2$.

be representable. When all of the structure including the prostar is representable, we obtain $*$-autonomous categories.