UNIVERSITY OF LEICESTER

Display calculi and nominal string diagrams

Samuel Balco

Supervised by Alexander Kurz and Tom Ridge

Thesis submitted for the degree of Doctor of Philosophy at the University of Leicester

Department of Informatics

December 2019

Declaration of Authorship

I, Samuel Balco, declare that this thesis titled, “Display calculi and nominal string diagrams” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Abstract

Department of Informatics

Display calculi and nominal string diagrams

by Samuel Balco

This thesis is divided into two sections, encompassing two main topics: display calculi and nominal string diagrams.

In the first section of the thesis, we introduce display calculi and present their advantages and drawbacks compared to sequent calculi. The rest of the section presents the calculus toolbox, a meta-tool for formalising display calculi. The tool includes a tree editor and a type-checker, which aid the user in exploring display calculi more efficiently.

Section two grew out of an attempt to build a calculus of simultaneous substitutions for a display version of first order logic. This section explores the topic of string diagrams, in particular, we present two categorical formalisations of nominal string diagrams, along with a formal translation of ordinary string diagrams into nominal string diagrams (and vice versa).

Acknowledgements

First and foremost, I would like to thank my boyfriend Olly, my parents and my sister for supporting me throughout my studies. Equally large is my gratitude to my first supervisor, Alex. He was an excellent supervisor with whom I spent countless hours pouring over category theory and many (if not more) talking about politics, art, technology, engineering or really anything we both found interesting. I would also like to thank his wife Lisa and his family, for the movie nights we enjoyed together. I want to thank his son Julius for allowing me to help him design his fort and my friend and colleague Drew for helping us in actually building it (see fig. 1).

A great thanks goes to my other supervisor Tom, who provided support and amusing commentary on life over countless coffee breaks.

I also want to thank all my friends and colleagues, in particular Claudia, who was an excellent house mate and fellow PhD student, Sam Jones, Frances McNally, Joe Turner and all my other Leicester friends, who have made Leicester and the UK my home. I would like thank all my collaborators and people I've met and had interesting conversations with throughout my PhD, listed alphabetically to avoid any accusations of favouritism: Sabine Frittella, Giuseppe Greco, Peter Jipsen, Drew Moshier, Alessandra Palmigiano, Paweł Sobociński, Georg Struth and Apostolos Tzimoulis. Finally, I would like to thank my examiners, Fabio Zanasi and Roy Crole for reading this thesis and providing such detailed comments and helpful references to relevant literature.

Figure 1: The Fort

Contents

Offer me a herbal tea and I will deliver a fart so dense and multi layered that your face will bend in on itself and become a butter dish

Bob Mortimer

1 Introduction

This thesis comprises of two parts, exploring the topics of tool support for display calculi and nominal string diagrams. Display calculi are a generalisation of sequent calculi [1] by Belnap [2] and have been studied extensively in the setting of modal and epistemic logics as well as other substructural logics (see [3–7]). They offer interesting proof-theoretic advantages over sequent calculi, such as the *cut*-elimination meta-theorem, along with better modularity/composability of different logics. However, this comes at the cost of verbosity, where a display version of a given logic will usually have a larger number of rules, compared to other formalisms like a sequent calculus or an axiomatic (Hilbert-style) system. We provide further details and examples of the advantages and disadvantages of display calculi in ch. 2, the introduction to part I of this thesis.

The following chapter (ch. 3) introduces and describes the features of the "calculus toolbox", a meta-tool for building and working with display calculi. This toolbox was developed to tackle the added complexity when working with display logics. We progressively developed several versions of the "calculus toolbox" testing different approaches and aims in each version.

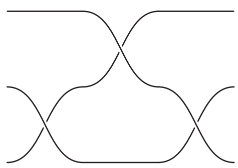
The toolbox described in ch. 3 is essentially a tree editor for constructing proofs in a display logic, which includes a typechecker that ensures correct proof tree construction. This toolbox grew out of a practical need to "test" display calculi by building proof trees. Due to the large number of rules, this was tedious and error prone to do by hand. The tool made this task much simpler by adding a visual proof tree editor and a simple way to export correct proofs as properly type-set LaTeX proof trees.

The calculus toolbox is a *meta*-toolbox, which means that the user can encode and modify their own display logic by adding inductively defined data-types for terms and encoding rules over these terms. These are in turn incorporated in the type-checker when constructing proof trees. We give further details on the internal workings of this tool in ch. 3. The toolbox described in this section is in fact a second iteration of the calculus toolbox, being a descendant of the original tool, presented in [8]. The second version made usability improvements aimed at mathematicians who are not necessarily programmers, by making it easier to build and modify display calculi within the toolbox.

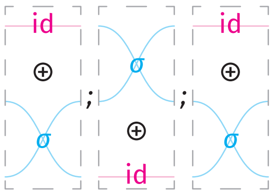
The next version of the toolbox, presented in ch. 4, focuses on extending the *meta*-toolbox to other logical formalisms besides display logics. As a result, the architecture of this tool changed drastically, compared to the previous version. The previous toolbox was built as *meta*-tool, which, given a description of a display logic and it's rules would be compiled into a tree editor with a type-checker, tailored for the specified logic. The new version of the toolbox takes a more unified approach, by combining the three languages for defining the terms, rules and proof trees into a single one.

In part II of this thesis, comprising of ch. 6 and ch. 7, we explore an altogether different topic of string diagrams. Whilst quite different to the work in part I, our study of string diagrams actually grew out of our work on display calculi. Inspired by a lecture series on graphical linear algebra using string diagrams, given by Paweł Sobociński at MGS 2017, we started exploring string diagrams as a way to formalise variable substitutions.

For example, given the following picture, it is quite easy to see that it represents a bijection:



Interpreting the picture as a function on ordered ports, the diagram above swaps port 1 and port 3 and leaves port 2 unchanged. This picture is not just intuitive, it can in fact be translated into "rigorous", i.e. algebraic notation, using two different multiplications for the horizontal and vertical composition of basic diagrams which make up the picture above. We decompose this diagram into into the picture below, which uses $\oplus$ for vertical composition and **;** for horizontal composition of basic diagrams. These include the straight **id**entity wire and σ, which represents crossing wires.



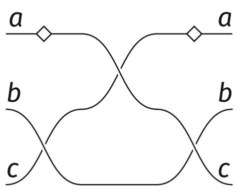
We thus obtain a 1-dimensional representation of the diagram above:

$$(\mathrm{id} \oplus \sigma) \ ; \ (\sigma \oplus \mathrm{id}) \ ; \ (\mathrm{id} \oplus \sigma)$$

The idea of using graphical syntax for mathematics in a rigorous way has been around for a long time. Whilst somewhat difficult to tell with certainty, arguably the first formal definition of string diagrams appears in the habilitation thesis of Günter Hotz [9]. However, forms of diagrammatic reasoning in areas such as knot theory have much earlier origins (see [10] for a nice historical summary). Definitions of string diagrams have also been introduced, amongst others, by Penrose [11], Joyal & Street [12, 13] and have cropped up in presentations of sequent calculi [14], linear logic as proof nets [15, 16], bigraphs [17], signal flow diagrams in control theory [18] and network theory [19] as well as in areas such as quantum physics and computing [20].

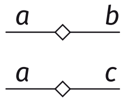
All of these formalisms are underpinned by the same category theory, namely that of (symmetric) monoidal categories, specifically **pro**duct and **p**ermuntation categories called PROPs for short, introduced by MacLane [21]. For an overview of classic/single sorted string diagrams see [22].

Returning to the graphical representation of a bijection above, we can see that whilst this diagram does represent a bijection, it does not represent the bijective substitution of names (permutations) that we want, since no names appear in the diagram. We can remedy this situation by adding labels (representing names) to the previous picture:

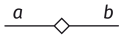


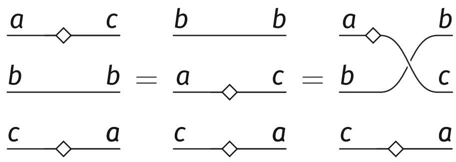
Now we can map a basic renaming $[a \mapsto b]$, which renames and a into a b, to a string diagram $\overset{a}{—}\!\diamond\!\overset{b}{—}$ (which we denote by δ_{ab}). We use the $\diamond$ symbol to explicitly denote the change of labels from a to b. We call this extended graphical formalism *nominal* string diagrams and in ch. 7, show that they are underpinned by the theory of nominal sets [23, 24], which provide an ideal framework for working with names.

Gaining expressiveness through the addition of names does come with a trade-off however. Because we want to use *nominal* string diagrams to present functions on names, the diagrams must of course follow the rules of being a function. Concretely, this means that we do not consider the following to be a valid diagram[1]:



In order to restrict to only "valid" diagrams, the vertical composition $\oplus$ becomes partial. The full details of the construction of *nominal* string diagrams with a partial $\oplus$, along with their categorical presentation via partially monoidal categories, are given in ch. 6.

Whilst ch. 6 mostly talks about nominal diagrams which deviate from ordinary string diagrams only by introducing named wires along with the basic "diamond" diagram  , in sec. 6.2.1, we present slightly simplified nominal string diagrams, which remove explicit twists σ. Our running example thus turns into several equivalent diagrams, bringing the graphical formalism further in line with the intended semantics of string diagrams representing functions on names:



Algebraically, we can write down the left most diagram above as $\delta_{ac} \oplus \mathrm{id}_b \oplus \delta_{ca}$.

Finally, ch. 7 focuses on streamlining nominal string diagrams further, by refining the notion of partial monoidal categories to nominal monoidal categories, thus providing a more complete categorical picture of the underlying structure of nominal string diagrams. Contents of this chapter were co-written with my supervisor, Prof. Alexander Kurz and have appeared as a paper at CALCO 2019, receiving the best paper award.

[1]The diagram is not a function because it maps the name a to two different outputs b, c.

I

Display calculi

Humans are not logical creatures. They are merely capable of doing logic.

T. M. O. Horne

I don't know if I've ever been to Australia... Have I been to Australia?

Justin Bieber

2

Background

In this chapter, we will give some background information on the kind of calculi we were considering when building the calculus toolbox and what design decisions we made as a result. More specifically, we give a brief introduction to Gentzen's sequent calculus and its generalization in the form of a display calculus, which is the primary formalism used to define calculi in the toolbox. Finally, we describe a further generalisation of display calculi to a multi-type setting.

2.1 Sequent calculus

To place the sequent and display calculi into context, we give a brief account of their history within the field of proof theory, which concerns itself with the study of proofs as mathematical objects. The roots of modern proof theory are often attributed to David Hilbert and his "Hilbert's program", which focused attention on mathematical proof as a formal object of mathematical study. Hilbert gave axiomatisations of numerous fields of mathematics, such as the foundations of geometry[1], algebraic number and mathematical logic, amongst others, in his later work using a framework of axiomatic schemas we now refer to as Hilbert calculi[2]. More precisely, a Hilbert calculus is a formal system of axiom schemas along with the *modus-ponens* derivation rule, where a formal mathematical proof (called a **derivation**) of

[1]Hilbert presented the axioms of geometry in his book **Grundlagen der Geometrie**.

[2]Whilst this formalism carries Hilbert's name, he was by no means the first or only person at the time, using such axiomatic calculi.

some logical statement P is a finite sequence of formulas ending in P, where each formula is either an instance of one of the axiom schemas, or has been derived via the application of the *modus ponens* rule from two previous formulas in the list. Given the axiom schema for the propositional fragment of classical logic:

$$A \to (B \to A) \tag{Ax 1}$$

$$(A \to (B \to C)) \to ((A \to B) \to (A \to C)) \tag{Ax 2}$$

$$(\neg A \to \neg B) \to ((\neg A \to B) \to A) \tag{Ax 3}$$

and the *modus ponens* rule:

$$\frac{A \qquad A \to B}{B} \tag{MP}$$

we can show the derivation of $P \to P$:

1	$(P \to ((P \to P) \to P))$	(Ax 1)
2	$((P \to ((P \to P) \to P)) \to ((P \to (P \to P)) \to (P \to P)))$	(Ax 2)
3	$((P \to (P \to P)) \to (P \to P))$	(MP) with 1 and 2
4	$(P \to (P \to P))$	(Ax 1)
5	$P \to P$	(MP) with 4 and 3

Whilst this formalism is extremely simple (having only one inference rule), it can be quite cumbersome to use for proofs of "regular mathematics", which are often conditional proofs of the form $\Gamma \vdash F$[3].

The sequent calculi **LK** and **LJ** were introduced by Gerhard Gentzen in 1935 [1] as formalisations of classical and intuitionistic versions of first order logic[4]. Gentzen's sequent calculus introduces a more complex formalism called a **sequent**, which has the form:

$$A_1, A_2, \dots, A_m \vdash B_1, B_2, \dots, B_n \qquad m, n \in \mathbb{N}$$

where As and Bs are formulas. Gentzen's sequent can be seen as analogous to the following formula in a Hilbert system:

$$A_1 \wedge A_2 \wedge \dots \wedge A_m \to B_1 \vee B_2 \vee \dots \vee B_n$$

where the turnstile is interpreted as implication and the comma as conjunction on the left and disjunction on the right of the turnstile.

Rather than having only one inference rule and many axioms, sequent calculi instead opt

[3]These can be read as "under the hypothesis Γ, F holds".

[4]However, in this section, we will only focus on the propositional fragments of these logics.

to have many rules of inference and only trivial axioms, such as

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

As a result, a proof or a derivation of some sequent $\Gamma \vdash \Delta$[5] is a tree, with $\Gamma \vdash \Delta$ at the root:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,Id \qquad \dfrac{}{B \vdash B}\,Id}{A \vee B \vdash A\,,B}\,\vee_L}{A \vee B \vdash B\,,A}\,P_R}{A \vee B\,,\neg B \vdash A}\,\neg_L$$

2.1.1 *Cut* rule and *cut*-elimination

When Gentzen introduced the sequent calculus, he showed that that it was sound and complete with respect to the semantics of first order logic. In order to show completeness, he included the *Cut* rule, which is an analogue to *modus ponens* in the Hilbert system:

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Sigma \vdash \Pi}{\Gamma, \Sigma \vdash \Delta, \Pi}\,Cut$$

However, he also showed that this rule could be eliminated from his calculus without any loss of expressiveness/deductive power. One reason for wanting to eliminate the *cut* rule is its non-*analyticity*, a consequence of the fact that the formula A only appears in the premises of the rule but not the conclusion. When performing proof search (searching for a proof by constructing a proof tree bottom-up) the *cut* rule presents an arbitrary choice, as the search procedure needs to pick some A to proceed. This makes the search space infinite, since there are infinitely many formulas A to choose from. Thus, being able to eliminate the use of *cut* is highly desirable, since it usually entails consistency of the given calculus [1].

2.1.2 Limitations

Because *cut*-elimination is an important property, many logics have been presented as sequent calculi. However, there are several logics for which there is no known *cut* free sequent calculus (also referred to as an analytic sequent calculus), e.g. the modal logic **S5** [25] or first-order Gödel logic [26] and calculi which provably have no analytic sequent calculus presentation [27]. These limitations have prompted generalisations of the sequent calculus, which do admit analytic calculi of the given logics. One example of a sequent calculus generalisation is the hyper-sequent calculus[6], which admits an analytic presentation of **S5** [26].

[5] Here Γ and Δ are arbitrary contexts, i.e. lists of formulas.

[6] Another formalism, which is at least as expressive as hyper-sequents [28] is the framework of display calculi.

Another reason for moving away from sequent calculi to display calculi is the *cut*-elimination proof itself. In his original paper, Belnap presented the *cut*-elimination meta-theorem for arbitrary display calculi. Whereas proofs of *cut*-elimination in sequent calculi are ad-hoc, Belnap proved that any display logic satisfying certain easily verifiable conditions on the rules, such as the aforementioned *analyticity*, would immediately enjoy the *cut*-elimination property.

In other words, a traditional *cut*-elimination argument for some sequent calculus essentially involves devising a bespoke algorithm for transforming any proof using the *cut* rule into a valid proof without it. Belnap's meta-theorem, on the other hand, gives a general-recipe for such a construction, irrespective of the logical connectives or rules[7] of the given display calculus.

2.2 Display calculi

The sequent calculus can be seen as an extension of the language of propositional formulas, by introducing two layers of terms. Instead of just having conjunction and disjunction at the level of formulas, we introduce a level of *structures* with the structural comma, which acts as a conjunction on the left and disjunction on the right of a turnstile. A natural question to then ask is, what about introducing *structural* counterparts for other connectives?

Indeed, extending implication and the truth values to the *structural* level by introducing *structural* counterparts to these *operational*[8] (formula) connectives, we obtain Belnaps's display calculus [2].

Structural	$>$		$<$		$,$		I	
Operational	$\succ\!-$	$\rightarrow$	$-\!\prec$	$\leftarrow$	$\wedge$	$\vee$	$\top$	$\bot$

Figure 2.1: *Structural* counterparts to *operational* connectives

In the **LK** sequent calculus, there is only one *structural* connective, with two introduction rules, corresponding to the reading of the "**,**" as a conjunction on the left and disjunction on the right of the turnstile[9]:

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ \wedge_L \qquad\qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee_R$$

However, no such direct "translation" happens with the implication, where A switches to the

[7] Provided they satisfy the conditions of the *cut*-elimination meta-theorem.

[8] The reason why the terminology of *structural*/*operational* connectives was chosen by Belnap [2] is not explained in his paper. The terms *operational rules* and *logical rules* are used interchangeably in the literature.

[9] These rules are slightly tweaked from the original formulation, however, they can be shown to be derivable from the classic rules of the **LK** calculus

other side of the turnstile:

$$\frac{\Gamma , A \vdash B , \Delta}{\Gamma \vdash A \to B , \Delta} \to_R$$

By including a new structural connective "$>$" in a display calculus, we can reformulate this rule to one more akin to $\vee_R$:

$$\frac{X \vdash A > B}{X \vdash A \to B} \to_R$$

As a result, the terms on both sides of the turnstile are no longer list of formulas, but trees of formulas. Also, notice that the *structural* connectives, sitting above the *operational* connectives in fig. 2.1 are grouped in pairs. Just like having the comma on the left and right of the turnstile means having either a conjunction or a disjunction, we use the **>** symbol for an implication on the right and a **co-**implication on the left.

Whilst the **co-**implication is not commonly used (for examples, see the **H-B** logic of [29] or paraconsistent propositional logics described in [30]), it arises analogously to the adjunction between the conjunction and implication on the preorder of propositions. Namely, because we have the implication as a right adjoint to the conjunction:

$$a \wedge b \vdash c \iff a \vdash b \to c$$

we can also formulate a **co-**implication as being the left adjoint to the disjunction:

$$a \rightarrowtail b \vdash c \iff b \vdash c \vee a$$

Indeed, adjunctions between the *operational* connectives are the core idea behind the "display" in display calculi. As shown above, these adjunctions can be formulated as reversible *display* rules:

$$(,/>)\,\frac{X , Y \vdash Z}{\overline{\overline{Y \vdash X > Z}}} \qquad \frac{Z \vdash X , Y}{\overline{\overline{X > Z \vdash Y}}}\,(>/,)$$

The term *display* comes from being able to move *structures* across the turnstile via these adjunctions. In this way, one can isolate/*display* any sub-*structure* on either the left or the right side of the turnstile. This is an important property of any display calculus and is referred to as the display theorem in Belnap's paper (Theorem 3.2 in [2]).

The display rules together with the display theorem are useful for several reasons; firstly, we gain a *cut*-elimination meta-theorem, similar to the *cut*-elimination arguments for Gentzen's sequent calculi, but applicable to a wider range logics (as mentioned earlier).

Secondly, the display framework imposes certain restrictions on the shape of the rules, which means that the interplay of different connectives via adjunctions and their *structural* properties, like commutativity or associativity, are given explicitly via *display* and *structural* rules. This makes display calculi much more modular, as it allows one to customize logics by adding new connectives and describing their interaction with other ones in a disciplined way. One can also produce *sub-structural* versions of existing logics by adding or removing *structural* rules (such as commutativity) for previously defined connectives.

2.3 Trade-offs

While there are clear advantages to display calculi in terms of modularity, the two levels of terms (*structural* and *operational*) together with the reversible *display* rules clearly introduce an overhead.

Having the two levels of terms also means having operational rules, which introduce *structural* counterparts of *operational* level connectives, e.g. for disjunction we have:

$$\vee_L \frac{A \vdash X \qquad B \vdash Y}{A \vee B \vdash X \,, Y} \qquad\qquad \frac{X \vdash A \,, B}{X \vdash A \vee B} \vee_R \tag{2.1}$$

All of these rules, taken together, result in large logic formalisms. Take, for example, the display version of classical logic without quantifiers, which has ~30 rules vs. just 18 for the **LK** sequent calculus without quantifiers. For bigger calculi like the **DEAK** (**D**ynamic **E**pistemic logic of **A**ctions and **K**nowledge) calculus [7], the number of rules quickly surpasses 100. As a direct consequence, the complexity and length of proofs increases. In the example below, the left and right proof trees are a display and sequent calculus proof of $((A \vee B) \wedge \neg B) \rightarrow A$ respectively:

$$\begin{prooftree}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$A \vdash A$}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$B \vdash B$}
\RightLabel{$\vee_L$}
\BinaryInfC{$A \vee B \vdash A \,, B$}
\RightLabel{$(>/,)$}
\UnaryInfC{$A > A \vee B \vdash B$}
\AxiomC{}
\RightLabel{$\bot_L$}
\UnaryInfC{$\bot \vdash \mathbb{I}$}
\RightLabel{$\rightarrow_L$}
\BinaryInfC{$B \rightarrow \bot \vdash (A > A \vee B) > \mathbb{I}$}
\RightLabel{$\boxed{\neg B}$-def}
\UnaryInfC{$\boxed{\neg B} \vdash (A > A \vee B) > \mathbb{I}$}
\RightLabel{$(,/>)$}
\UnaryInfC{$(A > A \vee B) \,, \boxed{\neg B} \vdash \mathbb{I}$}
\RightLabel{$Grishin_L$}
\UnaryInfC{$A > (A \vee B \,, \boxed{\neg B}) \vdash \mathbb{I}$}
\RightLabel{$\mathbb{I}W_R$}
\UnaryInfC{$A > (A \vee B \,, \boxed{\neg B}) \vdash A$}
\RightLabel{$(>/,)$}
\UnaryInfC{$A \vee B \,, \boxed{\neg B} \vdash A \,, A$}
\RightLabel{C_R}
\UnaryInfC{$A \vee B \,, \boxed{\neg B} \vdash A$}
\end{prooftree}
\qquad
\begin{prooftree}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$A \vdash A$}
\AxiomC{}
\RightLabel{Id}
\UnaryInfC{$B \vdash B$}
\RightLabel{$\vee_L$}
\BinaryInfC{$A \vee B \vdash A \,, B$}
\RightLabel{P_R}
\UnaryInfC{$A \vee B \vdash B \,, A$}
\RightLabel{$\neg_L$}
\UnaryInfC{$A \vee B \,, \neg B \vdash A$}
\end{prooftree}$$

> *Note.* $\boxed{\neg B}$ in the left tree is actually an abbreviation for the formula $B \rightarrow \bot$, since the display calculus version of classical logic presented here does not have negation as a primitive.

The fact that proofs in display logics are significantly longer than proofs in sequent calculus was in fact one of the driving reasons for building the calculus toolbox.

2.4 Modal logics and multi-type display calculi

In [31], the authors present a further generalisation of display calculi, by introducing a multi-type variant of **DEAK**. The extension presented in this paper is fairly simple, namely, they introduce formulas and *structures* at several distinct types. For modal logics, such as the **P**ublic **A**nnouncement **L**ogic [32], this is arguably already the case. Take the formula $\mathbf{K}_i\, A$ in **PAL**, which can be interpreted as "agent i knows A". Here, the type of the connective

$$\mathbf{K} : \mathcal{Ag} \times \mathcal{F} \rightarrow \mathcal{F}$$

takes two different types, $\mathcal{Ag}$ and $\mathcal{F}$, as arguments. We can therefore think of **PAL** as a (trivial) multi-type calculus by way of the following grammar:

$$\mathcal{F} \ni F ::= p \mid F \wedge F \mid F \vee F \mid \neg F \mid \mathbf{K}_i\, F \qquad p \in \mathcal{Al}, i \in \mathcal{Ag}$$

However, $\mathcal{Ag}$ is just a set of agents rather than inductively defined terms. We get a true multi-type calculus if the grammar has mutually recursive terms embedded in each other, like in this sample grammar adapted from [31]:

$$\begin{aligned} &\mathcal{Fm} \ni F ::= p \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \pi \twoheadrightarrow_0 F \mid \gamma \twoheadrightarrow_1 F \mid a \twoheadrightarrow_2 F \\ &\mathcal{Act} \ni \gamma ::= a \vartriangle \pi \qquad \pi \in \mathcal{Fn} \qquad a \in \mathcal{Ag} \qquad p \in \mathcal{Al} \end{aligned}$$

We can see that $\mathcal{Act}$ is an inductively defined set of terms much like $\mathcal{F}$, albeit with only one constructor (in this example).

You would not enjoy Nietzsche, sir. He is fundamentally unsound.

P. G. Wodehouse

3

Calculus toolbox

As we have seen in the previous section, display calculi generate an overhead in the number of rules compared to sequent calculi, which has a direct consequence on the size of the proof trees. This is in fact one of the major hurdles when working display logics, as building proof trees by hand or in LaTeX is time consuming and error prone. The Calculus toolbox, a program for defining and working with display calculi, aims to remedy this by providing a friendly user interface for defining display calculi and building proof trees which are well typed and exportable to LaTeX.

This chapter first gives a use case for the calculus toolbox, followed by a high-level overview of the graphical interface of the toolbox. Finally, we delve into some of the interesting implementation details of the front and back-end.

3.1 Muddy children puzzle

The calculus toolbox described in this thesis is actually the second version of the tool. The first version was presented in our paper [8] and we have since built an improved second version, described below. This tool is open source and is hosted on github[1].

In [8], we not only present the original tool, but demonstrate it's use in a "real world" setting, by formalising and proving correct the muddy children puzzle (for a good description and

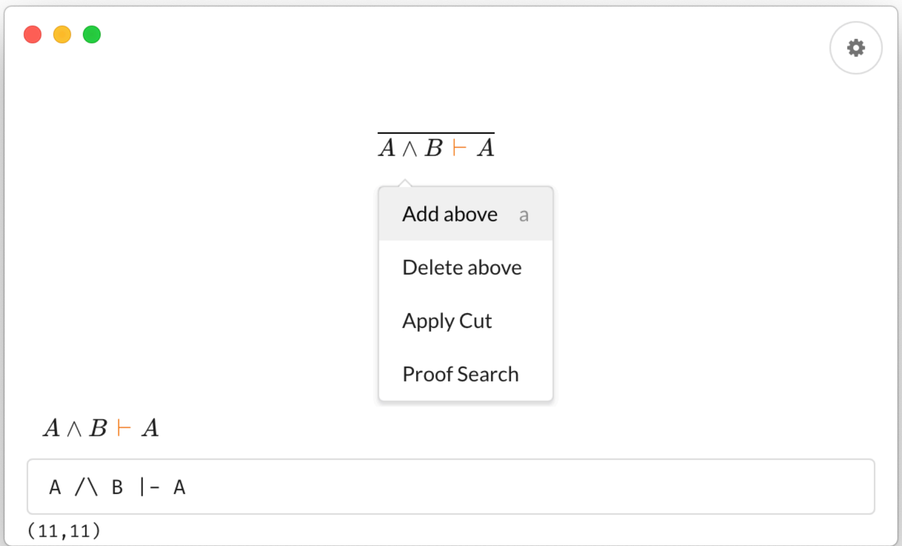
[1] https://github.com/goodlyrottenapple/calculus-toolbox-2

an informal sketch of a proof, see [33](Ch.1.1)). The statement of the puzzle and all the logical reasoning was done in **DEAK**, introduced in the previous chapter, and we fully formalised a version of this proof given in [34](Prop.24)[2]. As was already mentioned in sec. 2.3, proof trees in a display version of a logic can be much longer than in the equivalent sequent calculus. As a result, writing correct proof trees by hand or in LaTeX quickly becomes infeasible. For effect only, we've included the following snippet from our muddy children proof, which constitutes only a small part of the full proof:

Whilst this proof tree would be infeasible to typeset by hand, we only needed to provide the root of this proof tree to the calculus toolbox and then used the toolbox to construct the rest of the tree. Below, we give details on how this works, by describing the proof tree editor that is used to build such a tree.

3.2 Tree editor

In this section, we describe the main feature of the toolbox, the proof tree editor, pictured below.

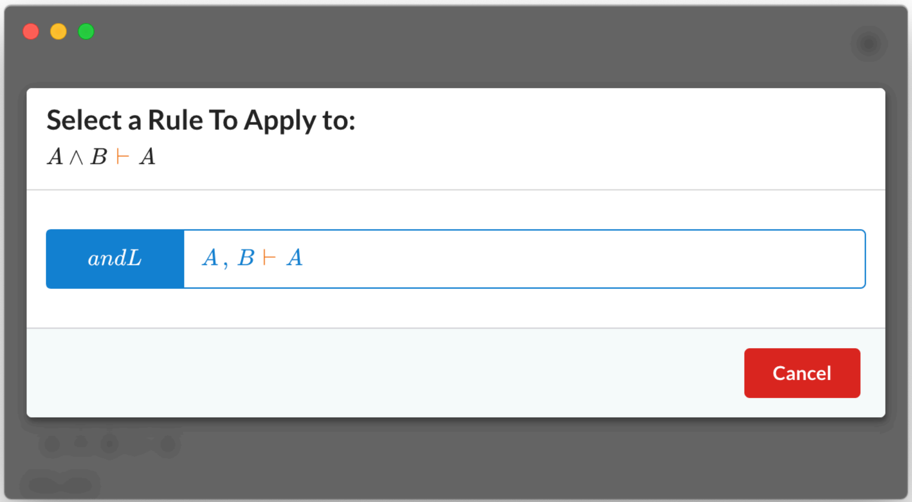


To start building a proof tree from the root to the leafs (bottom up), the user enters the sequent they wish to prove, using user defined ASCII syntax. The tree can then be modified,

[2]The formalised proof can be found at https://github.com/goodlyrottenapple/muddy-children

by clicking on any node in the tree and choosing to either delete the nodes above, add a new node, apply the *cut* rule (where the user is first prompted to enter the *cut* formula) or perform an automatic proof search.

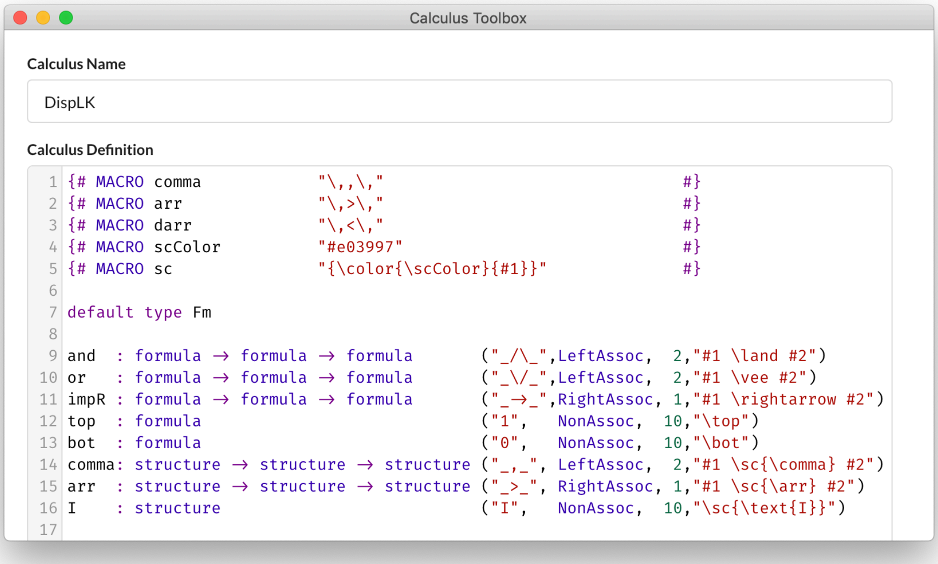
The proof tree editor acts as a proof assisntant, as selecting 'Add above' brings up a list of all rules applicable to the current node. In the example below, the tool lists all the applicable rules to the sequent $A \land B \vdash A$:



The proof search algorithm is a simple bounded depth first search which sequentially tries all applicable rules and backtracks if the depth limit is reached before a proof for the current branch is found. As it is undecidable whether or not a display calculus is decidable [3], the toolbox provides no guarantees that the proof-search will be successfull, and in fact the tool is only capable of findings simple proof trees, before the search space becomes too large.

3.3 Calculus editor

Because the calculus toolbox is meant to be an editor for arbitrary display calculi, the other major component of the toolbox is the calculus editor, which allows the user to define and edit display calculi. The user specifies the grammar of the formulas F and *structures* S of the logic in the **Calculus Definition** window.



The user defines each *operational*/formula connective and *structural* connective in a Haskell-like language. For example, the connective

$$\wedge : F \to F \to F$$

is encoded as

```
and : formula → formula → formula ("_/\_",LeftAssoc,2,"#1\land#2")
```

The additional parameters, given after the type signature, namely the ASCII parsing syntax `"_/\_"`, associativity `LeftAssoc`, fixity 2 and LaTeX syntax `"#1 \land #2"` are the used to generate the parser and pretty printer used in the tree editor window. The ASCII syntax is also used later to define the rules. For example, the rules 2.1 from the previous chapter are encoded as:

```
A |- X     B |- Y
----------------- orL ("\vee_L")
A \/ B |- X , Y

X |- A , B
----------- orR ("\vee_R")
X |- A \/ B
```

Unlike the original calculus toolbox, this version supports multi-type display calculi, described in sec. 2.4. As a result, the user must specify at least one default type (in this example it's `type Fm`). One can then introduce further types which can be given as parameters in the types of *operational*/*structural* connectives:

```
type Fn
trZer : formula{Fn} → formula → formula ( ... )
```

The snippet above is actually shorthand for the following full definition, where the unannotated formulas are associated with the given default type `Fm`:

```
trZer : formula{Fn} → formula{Fm} → formula{Fm} ( ... )
```

3.4 Internal representation

The original version of the toolbox used a JSON file to specify the syntax and rules of a display calculus and required the user to recompile the toolbox every time a change was made. This was a time consuming and brittle process, as it relied on calling several different tools and compilers (see fig. 3.1). To streamline this process, we rewrote the core of the tool in Haskell. This allowed us to simplify the definition language for the display calculi and instead of using JSON, we switched to a Haskell like syntax, described briefly above. We also wrote a custom parser generator and switched to a modular internal representation of sequents and trees, essentially writing an interpreter for display calculi which could be modified and updated at runtime. This greatly reduced the compilation speed (minutes vs less than a second) and made for a more robust system with better user error messages.

We use the following Haskell data type (shown slightly simplified) to encode any term of a user defined calculus:

```
data Term (l :: Level) (k :: TermKind) where
    Base :: Text → Term 'AtomL 'ConcreteK
    Meta :: SingI l ⇒ Text → Term l 'MetaK
    Lift :: SingI l ⇒ Term (Lower l) k → Term l k
    Con  :: (KnownNat n, SingI l, IsAtom l ~ 'False) ⇒
        Conn l n → Vec n (Term l k) → Term l k
```

This data type represents formulas and *structures*, which can either be concrete terms (e.g. $isSunny \rightarrow \neg isRaining$) or meta variables, which appear in rules, like A or X in 2.1. In full detail:

- The **`Base`** constructor is used for building concrete atoms, like $isSunny$. Internally, the tool uses strings to represent atoms.
- The **`Meta`** constructor is used for meta variables, used in the definitions of rules.
- The **`Lift`** constructor promotes an atom to a formula or a formula to a *structure*. This is usually done implicitly in the informal descriptions of the grammar and the toolbox can automatically parse and deduce the appropriate level of all terms.

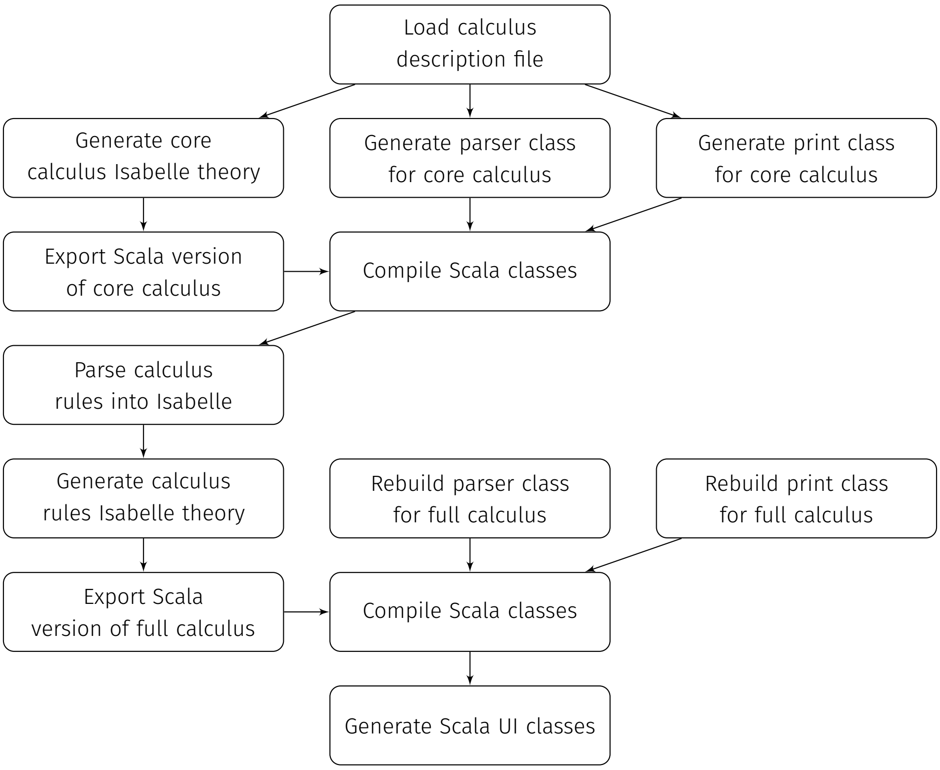


Figure 3.1: Original toolbox compilation process

- The **Con** constructor encodes connectives of arbitrary arity. It uses a length indexed vector type to ensure that the connective with a given arity is given the right number of arguments as sub-terms.

3.5 Type checking

Due to the introduction of types, for multi-type display calculi, we need to type-check terms to ensure that atoms are not assigned two different types and that multi-typed connectives are given arguments of the correct type. For example, the **DEAK** calculus contains the following connective:

$$\Delta_0 : F_{\mathcal{F}n} \to F_{\mathcal{F}m} \to F_{\mathcal{F}m}$$

If we try to type check the term $f \,\Delta_0\, (a \wedge f)$, we get an error, since $\wedge$ has the type $F_{\mathcal{F}m} \to F_{\mathcal{F}m} \to F_{\mathcal{F}m}$, so naturally the unification of f at type $F_{\mathcal{F}n}$ and $F_{\mathcal{F}m}$ fails.

When parsing the rules of the calculus, the type-checking algorithm is also used to disambiguate the level of meta-variables. Consider the following rule:

$$\frac{X \vdash A > B}{X \vdash A \to B}$$

Knowing that $\to$ is an operational/formula connective, we know that the variables A, B

can only be substituted with formulas, whereas X can be an arbitrary *structure*. Instead of having to specify this explicitly, the toolbox can infer this information during type-checking, by keeping a track of the context the meta variables appear in. In the premise of the rule, X, A and B appear in the context of *structural* connectives. However, in the conclusion, A and B get "downgraded" to formula variables, because they appear as arguments to $\wedge$. After type-checking, this information is used to adjust the meta variables accordingly. For special rules like the *Id* rule

$$\frac{}{a \vdash a}\, Id$$

where we want to stipulate that a is an atom, rather than a *structure* or a formula, we can explicitly declare an atom meta-variable by prefixing the variable name with `at_`:

```
------------ Id
at_a |- at_a
```

3.6 Front-end

Unlike the back-end, which is written in Haskell, we opted to use JavaScript, namely Electron and React, for the front-end. The reasoning behind this decision was to try to provide uniform interface across all the major platforms as well as to potentially make the tool available online, without the need to download anything. These two considerations made HTML and JavaScript the ideal candidates to use when building the UI.
To link the front and back-end together, we used a REST API generator framework Servant to generate an interface the front-end can use to communicate with the Haskell back-end. This includes functionality such as parsing of user input into display sequents and running proof search or type-checking of a proof tree built by the UI editor.

The use of Servant allows one to define REST APIs using Haskell's type-system, ensuring type safety of an implementation with regards to the specified API. Servant also automatically generates a JavaScript boilerplate library, built for the defined API, which can be plugged into the front end to ensure that communication between the front and back-end is implemented correctly. For example, the type `API` below describes a REST API end-point `parseFormula`, which takes a raw `ParseTerm` string, calls the parser and returns a parsed formula `Term`, wrapped together with its LaTeX type-setting information inside a `LatexTerm`:

```
type API =
```

```
    "parseFormula" :>
    ReqBody '[JSON] ParseTerm :>
    Post '[JSON] (LatexTerm (Term 'FormulaL 'ConcreteK))
```

The Servant library then generates the following boilerplate JavaScript code from the given type:

```
postParseFormula = function(port, body, onSuccess, onError) {
  var xhr = new XMLHttpRequest();
  xhr.open('POST', `http://localhost:${port}/parseFormula`, true);
  ...
  xhr.send(JSON.stringify(body));
};
```

This function is then used by the front-end to pass raw user input to the Haskell back-end for parsing. The back-end implements the API functionality by defining the function `parseFormulaHandler`:

```
parseFormulaHandler :: ParseTerm →
    AppM r (LatexTerm (Term 'FormulaL 'ConcreteK))
parseFormulaHandler ParseTerm{..} = ...
```

As the type of this function suggests, the back-end implementation operates on native Haskell data-types (i.e. `ParseTerm`), rather than raw JSON. The translation of the request body and the response is again handled automatically by the Servant library.

3.7 Limitations

Whilst the second version of the toolbox introduced new features like the ability to define multi-type display calculi and streamlined the process of recompiling calculi, there are certain limitations which became apparent when trying to use the tool to formalise a display version of first order logic with quantifiers **DFOL**[35].

Due to the nature of **DFOL**, the simple type system used in the tool was not powerful enough to properly describe its connectives. The toolbox also imposes a rigid hierarchy of definitions, i.e. we have 4 levels of terms each nested in the next:

$$\begin{array}{ccccccc} \text{Atom} & & \text{Formula} & & \text{Structure} & & \text{Sequent} \\ a & \hookrightarrow & a \wedge b & \hookrightarrow & a \wedge b, c \vee d & \rightarrow & a \wedge b, c \vee d \vdash e \end{array}$$

This hierarchy makes sense when describing many display logics, however there is no reason why the toolbox could not be used to build and work with terms of different calculi, not necessarily following the same format.

Both of these limitations have motivated another version of the toolbox, presented in the next section.

Object-oriented programming is an exceptionally bad idea which could only have originated in California.

Edsger Dijkstra

4

Toolbox **t3**

BEFORE we introduce the third iteration of the calculus toolbox, we briefly describe the **DFOL** calculus, a display version of first order logic, which guided the design decisions of the third version of the calculus toolbox.

4.1 FOL displayed

As discussed in ch. 2, the main idea behind display logics is to give *operational* (formula) connectives a *structural* counterpart and introduce display rules which encode the notion of an adjunction between different connectives. The display version of FOL presented in [35] (Chapter 4) extends this notion to the universal and existential quantifiers of first order logic. There is ample literature which explores a modal operator-like interpretation of these quantifiers, e.g. [36–38]. The formalisation of **DFOL** in [35] follows the categorical approach to quantifiers as adjoints, which is described in detail by Lawvere in [39, 40].

In this chapter, we mainly focus on how to define the universal quantifier of **DFOL** in the calculus toolbox. In order to do this, we first extend fig. 2.1 with the following *operational* and *structural* connectives to obtain a small fragment of **DFOL** containing ∀ and ∃:

Structural symbols	(x)		Qy		$[\![\vec{t}]\!]$	
Operational symbols	$\circ_x$	$\circ_x$	$\exists y$	$\forall y$	$[\vec{t}]$	$[\vec{t}]$

Figure 4.1: *Structural* and *operational* connectives of **DFOL**

Compared to the standard Gentzen calculus, we have two additional *operational* connectives (and their *structural* counterparts):

- $\mathrm{Q}y\, A$ denotes $\exists y\, A$ on the left of the turnstile and $\forall y\, A$ on the right of the turnstile.
- $\circ_x\, A$ denotes a fresh variable x in the formula A
- $[\vec{t}]\, A$ represents a list of simultaneous substitutions $\vec{t}$ applied to F, where each term $t_m \in \vec{t}$ is associated to a free variable v_m in A.

In the usual Gentzen calculus, substitution is usually treated as a meta-operation in the introduction rules, e.g.:

$$\forall_L\ \frac{A[t/x]\,,\ \Gamma \vdash \Delta}{\forall x\, A\,,\ \Gamma \vdash \Delta} \qquad\qquad \frac{\Gamma \vdash A[y/x]\,,\ \Delta}{\Gamma \vdash \forall x\, A\,,\ \Delta}\ \forall_R$$

In these rules, the term $A[t/x]$ is not a syntactic object, but rather some A' obtained by substituting t for the free variable x in A. However, $[\vec{t}]\, A$ is a "first class" term in **DFOL**, which essentially means we internalise the operation of substitution into the calculus.

Our display version of FOL also differs from most other formalisations in that it is a multi-type display calculus, which means that the formulas and structures are tagged with types in a similar way to the terms of **DEAK**. However, unlike in the case of **DEAK**, were we only had 5 types ($\mathcal{F}m$, $\mathcal{A}ct$, $\mathcal{F}n$, $\mathcal{A}g$, $\mathcal{A}l$), the number of types in **DFOL** becomes infinite.

This is due to the fact that we take all subsets of free variables as types, giving a formula the type corresponding to the set of it's free variables:

$$f(x,y) : F_{\{x,y\}} \qquad\qquad \forall x\, \forall y\, f(x,y) : F_{\varnothing}$$

The new connectives of the calculus thus become heterogeneously typed:

$$\forall y : F_{X \uplus \{y\}} \to F_X$$

The type signature of the universal quantifier also implicitly places a side-condition on the formula A in $\forall x\ A$. Since A has to have a type $F_{X \uplus \{x\}}$, x must appear in the set of free variables of A (since the type of a formula is always tagged with it's free variables) and becomes bound/hidden when quantified over by $\forall$. We can reformulate the type signature above to make this more explicit:

$$\forall y : F_X \to F_{X \setminus \{y\}} \text{ with } y \in X$$

Already, we can see that the type signatures of **DFOL** are much more complex than those of **DEAK**. Besides needing a richer type-system for describing side conditions like $y \in X$, the type signatures above are also dependently typed, that is, the type of the argument to $\forall y$ depends on the given y. The universal quantifier can thus be seen as a binary connective, which takes a variable y and a formula, where y must appear as a free variable:

$$\forall : (y : \mathcal{V}ar) \to F_{X \uplus \{y\}} \to F_X$$

4.2 Dependent types

Trying to implement the definition above presented interesting challenges. Because the quantifiers have dependent types, we first tried to formalise the calculus in Agda and came up with this naive definition for the ∀ quantifier :

```
data F : FSet ℕ → Set where
  ∀ : {𝓃 : FSet ℕ} → (n : ℕ) → F (insert n 𝓃) → F 𝓃
```

However, this definition contains a subtle bug, wherein the type `insert n X` does not actually preclude X from containing n, which is what we want i.e. we can only quantify over free variables that actually appear in a formula. The resulting type of a quantified term should then remove n from the set of free variables. The following definition is in fact the correct one:

```
data F : FSet ℕ → Set where
  ∀ : {𝓃 : FSet ℕ} → (x : ℕ) → {_ : isElem x 𝓃} →
      F 𝓃 → F (remove n 𝓃)
```

Unfortunately, this definition is still difficult to work with, as it does not take into account the fact that a list $[y, x]$ and $[x, y]$ represent the same underlying set $\{x, y\}$. Once we define a notion of set equality for lists ($\simeq$), which is different to the usual syntactic equality, we arrive at this final definition:

```
data F : FSet ℕ → Set where
  ∀ : {𝓃 𝓃₂ : FSet ℕ} → (x : ℕ) → {_ : isElem x 𝓃} →
      F 𝓃 → {_ : 𝓃₂ ≃ remove x 𝓃} → F 𝓃₂
```

Defining ∀ this way, we get a flexible enough definition to work with in Agda. However, in practice, this formalisation is still cumbersome due to the following limitations:

- there is no basic data-type of (finite) sets for arbitrary types in Agda and formalising and working with finite sets of names seems like an unnecessary overhead.
- Agda cannot (usually) infer implicit arguments like `{_ : isElem x 𝓃}` and to set up things in such a way that these arguments are automatically discovered takes a lot of experience with dependent types and Agda.

These two limitations were the driving motivation behind **t3**, the third iteration of the calculus toolbox. Before delving into details, here is a definition of the ∀ quantifier in **t3**[1]:

```
data F : Set Name → Type where
  ∀ : {𝓃 : Set Name} → {𝓃₂ : Set Name} →
      (x : Name) → [ x ∈ 𝓃 ] → F 𝓃 → [ 𝓃₂ ≡ 𝓃 \\ x ] → F 𝓃₂
end
```

In the following section, we will describe the language **t3** uses, explaining the details of the definition above.

4.3 t3 core

At its core, **t3** is a version of a dependently typed λ-calculus without the λ, that is, there is no λ-abstraction or β-reduction in the calculus. The only terms one can construct are applications of non-reducible terms, namely type constructors. The language does have application and function types, as these are needed to define the type constructors of terms. In the example above, we see three different Π/function types.

The curly braces in the definition above denote implicit arguments, which can be given a name, such as `{𝓃 : Set Name}`. Implicit arguments will be depended upon by another type and can often be inferred from this type without the need to be supplied explicitly by the user.

The `Prop` type, denoted by square brackets encodes side-conditions, such as $x \in N$ and is not a dependent type. `Prop`s are a subset of types in **t3**, translatable into an SMT solver theory. Details of this translation are provided in the following section. Finally, we have the explicit Π-type which can also be bound to a name and referred to by terms under the Π. Below, we describe the internal representation of terms of **t3**, which is again written in Haskell.

> *Note.* We used Löh et al.'s [41] excellent tutorial, describing how to implement a dependently typed lambda calculus, as a starting point. The data-type representing **t3** terms is an extension of the one given in the tutorial.

```
data Term = StarT
          | PropT
          | NameT
          | MkName Text
          | SetT Term
          | MkSet Term [Term]
          | IntT
```

[1] Perhaps confusingly, Agda's `Set` is now `Type`, since we wanted to use `Set` for finite sets in **t3**.

```
            | MkInt Int
            | Π (Maybe Text) Term Term
            | IΠ Term Term
            | Term :⇒: Term
            | Bound Int
            | Free Name
            | Term :@: [ExplImpl Term]
```

There are 5 base/built-in types, listed below both in **t3** syntax and the corresponding Haskell core `Term`:

t3	Haskell	
`* / Type`	`StarT`	the type of all types[2]
`Prop`	`PropT`	the type of propositions/predicates, decidable in an SMT solver (essentially the `Bool` type)
`Name`	`NameT`	the type of names, used for variables or constants like *x* or *isSunny*
`Set a`	`SetT a`	the type of finite sets, with built-in set equality
`Int`	`IntT`	the type of integers (represented by Haskell's `Int`)

Because **t3** is dependently typed, there is no separation of terms and types, as one can refer to types in terms and vice-versa. We use de Bruijn indices for binders, replacing a named variable with a bound index in the internal representation:

t3	Haskell
`{a : *} → Set a`	`IΠ StarT (SetT (Bound 0))`

4.4 SMT solvers and the `Prop` type

As we have seen in the previous section, the Agda and **t3** data-type definitions of the ∀ quantifier are almost identical. In this section we focus on the encoding of the side-condition $y \in \mathcal{N}$, which appears in the type of ∀. In Agda, we encode this side condition as an implicit argument `{_ : isElem x` $\mathcal{N}$`}`, where `isElem` is itself a regular Agda type encoding finite set membership:

```
data isElem {A : Set} : A → List A → Set where
  here  : ∀ {x y : A} {xs : List A} → y ≡ x → isElem y (x :: xs)
  there : ∀ {x y : A} {xs : List A} → isElem y xs → isElem y (x :: xs)
```

[2]We do not have a type hierarchy like in Agda and admit `Type : Type`

However, this turns out to be rather impractical if we want to construct any concrete terms using the ∀ quantifier, as Agda cannot automatically construct the proof/term of the `isElem` type:

> **Example 4.1.** If we want to quantify over a term with a set of free variables $\{1, 2, 5, 7\}$, specifically, quantifying over the variable 5, we need to prove that 5 appears in the list $[1, 2, 5, 7]$[3].
>
> Thus, we need to construct a term of type `isElem 5 (1 :: 2 :: 5 :: 7 :: [])`, which is `there (there (here refl))` (the `refl` constructor witnesses the fact that `5 ≡ 5`).

We could do better by leveraging a "trick" called *proof by reflection*, described in [42]. Rather than encoding a property such as elementship in a data-type, we can define a type level function `isElem : ℕ → List ℕ → Set` which reduces to the type ⊥ if the element does not appear in the list and ⊤ otherwise:

```
isElem : ℕ → List ℕ → Set
isElem x [] = ⊥
isElem x (y :: xs) with x ≟ y
isElem x (y :: xs) | yes _ = ⊤
isElem x (y :: xs) | no  _ = isElem x xs
```

We can make use of the fact that Agda performs β-reduction when elaborating terms and given an element and a concrete list it appears in, `isElem?` will evaluate to ⊤. Because of the way ⊤ is defined, Agda will in fact be able to infer the value of type ⊤[4].

Whilst *proof by reflection* is a powerful technique, we believe it is also much more difficult to engineer than the approach we chose in our tool. Instead of proving propositions like $x \in \{z, x, y\}$ by directly encoding them as data-types or using *proof by reflection*, **t3** leverages the power of SMT solvers to automate away such proofs completely.

Given a proposition of a certain type, **t3** translates it into SMT-LIB, which is a language for interfacing with theorem provers such as CVC4 or Z3, and passes it to the CVC4 solver as a constraint. If, after collecting all the constraints during type-checking, the SMT solver returns "satisfiable", the type-checking succeeds. Otherwise, the SMT solver returns the subset of constraints which are unsatisfiable as an error. We give the propositions translatable into SMT the type `Prop`[5] in **t3**.

To make this approach as flexible and modular as possible, we only provide direct translation to and from SMT-LIB for the built in types `Int`, `Name` and `Set`. We then provide three

[3]Finite sets are encoded as lists.

[4]There is in fact only one unique value of type ⊤ and Agda knows this, therefore it will always automatically infer this value.

[5]Not to be confused with a PROP, which is an entirely different concept introduced in ch. 7.

mechanisms that allow the user to extend this translation to their theories.

The simplest way to interface with the SMT solver is via the `smt-builtin` command. This command acts as a wrapper for importing built-in SMT theory functions. For example, CVC4 includes a theory of finite sets[6], with definitions of functions and predicates like union, intersection, membership, etc. To import the set membership predicate into **t3**, we write:

```
smt-builtin (∈) [ member ] : {a : Type} → a → Set a → Prop end
```

In the code above, we first supply the name of the function/predicate as we want it to appear in our **t3** theory (∈), followed by the name of the function/predicate as it appears in the SMT-LIB library (in the square brackets). Because **t3** is strongly typed, we have to also include the type of the function/predicate we are importing, in this case `{a : Type} → a → Set a → Prop`.

> *Note.* **t3** does not check that the type given in the interface actually matches the type inside the SMT solver and one needs to consult the documentation of the SMT solver theories to make sure that the type signatures match.

If we want to define more complex functions/predicates, we can use the second available mechanism `smt-def`:

```
smt-def (∉) : {a : Type} → (x : a) → (X : Set a) → Prop where
  (not (elem x X))
end
```

Here we introduce the negated set membership predicate, where the body of the definition `(not (elem x X))` is an SMT-LIB expression. There are some minor differences in the dialect of lisp used in **t3** vs. the SMT-LIB lisp dialect, the main of which is the use of custom syntax for atoms/keywords. **t3** can automatically infer if a name refers to a variable or a constant, however, prefixing a name with `'` makes it explicit that the given name is a constant[7]. This extension is a compromise in the way **t3** parses definitions like `(= a b)`, which would otherwise be awkward to parse due to the fact that = is a reserved keyword and has special parsing rules elsewhere in the language. We can, circumvent the default rules for = by writing `('= a b)` in **t3**.

Finally, **t3** also allows simple data-type[8] lifting via the `smt-data` command:

[6] For the list of available functions and predicates over finite sets in CVC4, see: http://cvc4.cs.stanford.edu/wiki/Sets

[7] We could have also written `('not ('elem x X))`.

[8] Specifically, polymorphic algebraic data-types not containing any dependent types.

```
smt-data List : Type → Type where
    ∅ : {a : Type} → List a
  | (;) : {a : Type} → (hd : a) → (tl : List a) → List a
end
```

> *Note.* When defining a lifted data-type, the user must provide names for all the arguments of each constructor[9]. This requirement stems from the way data-types are defined in an SMT solver.

Once the data-type definition has been lifted, it can be used in other `smt-def`s. CVC4 contains powerful features such as the ability to write recursive function definitions over user defined data-types, which can then be used in type-checking **t3** programs. For an example of this, see app. A.

4.5 Translation to LaTeX

As we have re-iterated throughout this chapter, the main purpose of building the calculus toolbox is to build (proof) trees. More specifically, our initial motivation was to build proof trees for display calculi. With **t3**, we relaxed the structure of trees to be arbitrary algebraic data-types, such as Gentzen's sequents, made up of first order formulas. Because **t3** is dependently typed and supports GADTs[10], we can encode proof trees as data-types directly (see app. A for a full example).

Given the following rules of the sequent calculus:

$$Id \, \frac{}{a \vdash a} \qquad\qquad \frac{\Gamma, A \vdash \Delta}{\Gamma, A \land B \vdash \Delta} \, \land_{L1}$$

we can encode the inference rules as data constructors of the "derivable" data-type $\vdash$:

```
data (⊢) : List F → List F → Type where
    Id : {a : Name} →
      ------------------------
      (At a) ; ∅ ⊢ (At a) ; ∅

  | AndL1 : {Γ : List F} → {Δ : List F} → {A : F} → {B : F} →
         A ; Γ ⊢ Δ
    → ---------------
      (A ∧ B) ; Γ ⊢ Δ
```

We can then build valid proof trees as `def`initions in **t3**:

[9] This is similar to defining a record type in Haskell.
[10] Generalised algebraic data-types

```
def pt : At 'a v At 'b ; ∅ ⊢ At 'a v At 'b ; ∅ where
  CR (OrL
        {∅} {∅}
        {At 'a v At 'b ; ∅} {At 'a v At 'b ; ∅}
        (OrR1 Id) (OrR2 Id))
end
```

To allow for incremental construction of proof trees in a fashion, similar to the previous versions of the calculus toolbox, **t3** allows the user to build partial terms using "holes" (much like in Agda). For a definition like the one above, the user would generally first define the type of the term and leave the body undefined by indicating a hole with `?`.

```
def pt : At 'a v At 'b ; ∅ ⊢ At 'a v At 'b ; ∅ where
  CR ?
end
```

Elaborating the `def`inition above, **t3** infers the type of the hole:

```
================================
?0 : At 'a v At 'b ; ∅ ⊢ At 'a v At 'b ; At 'a v At 'b ; ∅
================================
```

This way we can produce the proof tree step by step, using **t3** as a guide. Once we build the proof tree, the final feature of **t3** allows us to pretty print it in LaTeX or in fact any other user-defined syntax, once we provide **t3** with a translation function:

```
language LaTeX

translation (⊢) to LaTeX where
    Id : x ⊢ y → "\AXC{}\RightLabel{$Id$}\n\UIC{$#{x}\vdash#{y}$}"
end
```

The `translation` definition allows simple pattern matching on all the constructors of a given data-type and uses basic string interpolation to place arguments `x` and `y` into the string on the right-hand side of the pattern.

> *Note.* **t3** tries to recursively find and apply the translations to all nested terms within a data-type. If such translation was not defined, it defaults to outputing **t3** syntax.

Once we have defined the translation for all the given data-types, we can write `translate pt to LaTeX end`, which produces:

```
\AXC{}\RightLabel{$Id$}
\UIC{$\cons{a}{} \vdash \cons{a}{}$}\RightLabel{$\vee_{R1}$}
\UIC{$\cons{a}{} \vdash \cons{a \vee b}{}$}
\AXC{}\RightLabel{$Id$}
\UIC{$\cons{b}{} \vdash \cons{b}{}$}\RightLabel{$\vee_{R2}$}
\UIC{$\cons{b}{} \vdash \cons{a \vee b}{}$}\RightLabel{$\vee_{L}$}
```

```
\BIC{$\cons{a \vee b}{} \vdash \cons{a \vee b}{\cons{a \vee b}{}}$}
\RightLabel{$C_R$}
\UIC{$\cons{a \vee b}{} \vdash \cons{a \vee b}{}$}
```

Typeset in LaTeX, this produces the following proof tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{a \vdash a}\,Id}{a \vdash a \vee b}\,\vee_{R1} \qquad \dfrac{\dfrac{}{b \vdash b}\,Id}{b \vdash a \vee b}\,\vee_{R2}}{a \vee b \vdash a \vee b, a \vee b}\,\vee_L}{a \vee b \vdash a \vee b}\,C_R$$

II

Nominal string diagrams

They want you to believe the Sun is hot. I urge you to ask yourself 'Have they ever touched it?' Think about it.

Jaden Smith

[redacted].

[redacted]

5

Introduction

OUR motivation for introducing nominal string diagrams has been formulated by the slogan ‘only connectivity matters’[1]. In the ordinary setting, this is achieved by ordering input and output wires of string diagrams and using their ordinal numbers as implicit names. We write $\underline{n} = \{1, \dots n\}$ to denote the set of n numbered wires and $f : \underline{n} \to \underline{m}$ for diagrams f with n inputs and m outputs. On the other hand, if only connectivity matters, it is natural to consider a formalisation of string diagrams in which wires are named, rather than ordered. Thus, instead of ordering wires, we fix a countably infinite set $\mathcal{N}$ of ‘names’ $a, b, \dots$, on which the only supported operation or relation is equality.

In this part of the thesis, we explore two approaches to giving a formal (categorical) definition to named string diagrams. In ch. 6, we define a general notion of partial monoidal categories, which allow us to account for the use of named wires.
The following ch. 7 focuses more closely on the ‘nominal’ aspect of nominal string diagrams by presenting them as categories internal in the category of nominal sets, introduced by Gabbay and Pitts [23, 24, 43].

Calculus of simultaneous substitutions

As we already mentioned in ch. 1, the driving motivation behind formalising nominal string diagrams was to develop a calculus of simultaneous substitutions. The advantages of a 2-dimensional calculus for simultaneous substitutions over a 1-dimensional calculus are the following.

[1] see [20] sec. 10.1

A calculus of substitutions is an algebraic representation, up to isomorphism, of the category $\underset{\sim}{\mathsf{n}}\mathbb{F}$ of finite subsets of $\mathcal{N}$. In a 1-dimensional calculus, operations $[a{\mapsto}b]$ have to be indexed by finite sets S

$$[a{\mapsto}b]_S : S \cup \{a\} \to S \cup \{b\}$$

for sets S with $a, b \notin S$.

On the other hand, in a 2-dimensional calculus with an explicit operation $\uplus$ for set union, indexing with subsets S is unnecessary. Moreover, while the swapping

$$[a{\mapsto}b, b{\mapsto}a] : \{a, b\} \to \{a, b\}$$

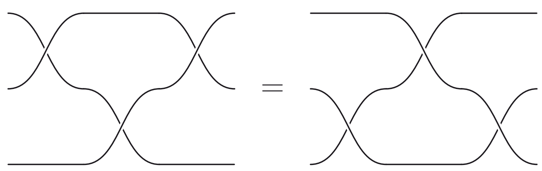
in the 1-dimensional calculus needs an auxiliary name such as c in $[a{\mapsto}c]_{\{b\}}$; $[b{\mapsto}a]_{\{c\}}$; $[c{\mapsto}a]_{\{b\}}$ it is represented in the 2-dimensional calculus directly by

$$[a{\mapsto}b] \uplus [b{\mapsto}a]$$

Finally, while it is possible to write down the equations and rewrite rules for the 1-dimensional calculus, it does not appear as particularly natural. In particular, only in the 2-dimensional calculus, will the swapping have a simple normal form such as $[a{\mapsto}b]\uplus[b{\mapsto}a]$ (unique up to commutativity of $\uplus$).

Symmetries in the nominal setting

From a graphical point of view, the move from ordered wires to named wires means that we no longer need to consider wire-crossings, or more techincally, there are no symmetries to take care of. This can simplify the rewrite rules of calculi formulated in the named setting. For example, rules such as



are not needed anymore. For more on this compare figs. 7.3, 7.4.

For a geometric intutition, there should be a result analogous to the note after Theorem 3.12 in [44], where ismorphism of nominal string diagrams can be seen as equivalent to ambient isotopy in 4 dimensions (also see Chapter 3 in [13]), though we have not investigated this fully.

Partial commutative vs total symmetric tensor

One reason why ordered names/wires are convenient is that the tensor $\oplus$ is given by the categorical coproduct (addition) in the skeleton $\underset{\sim}{\mathbb{F}}$ of the category of finite sets $\underset{\sim}{\mathsf{n}\mathbb{F}}$. Even though $\underline{n} \oplus \underline{m} = \underline{m} \oplus \underline{n}$ on objects, the tensor is not commutative but only symmetric, since the canonical arrow $\underline{n} \oplus \underline{m} \to \underline{m} \oplus \underline{n}$ is not the identity.

On the other hand, in the category $\underset{\sim}{\mathsf{n}\mathbb{F}}$ of finite subsets of $\mathcal{N}$, there is a commutative tensor $A \uplus B$ given by union of disjoint sets. The feature that makes commutativity possible is that $\uplus$ is partial with $A \uplus B$ defined if and only if $A \cap B = \varnothing$.

In order to define these partial tensors in the context of category theory, ch. 6 introduces the notion of partial monoidal categories. The chapter then goes on to define a specific partial monoidal category, corresponding to the 2-dimensional calculus of simultaneous substitutions. The calculus we propose is given in fig. 6.2 and we prove it sound and complete w.r.t. $\underset{\sim}{\mathsf{n}\mathbb{F}}$.

Overview

To summarise, sec. 6.1 introduces partially monoidal categories, sec. 6.2 defines the syntax and semantics of our language of named string diagrams and sec. 6.3 and sec. 6.4 show completeness of the axiomatisations of, respectively, bijections and functions. sec. 6.5 gives a short account of the software we developed to support the mathematical reasoning of this chapter.

Giving an alternative account of partial tensors, sec. 7.2 develops the notion of a monoidal category internal in another (monoidal) category. sec. 7.3 is devoted to examples, while sec. 7.4 introduces the notion of a nominal PROP, sec. 7.5 shows that the categories of ordinary and of nominal PROPs are equivalent and sec. 7.6 provides a way of translating ordinary string diagrams into nominal ones and vice versa.

καὶ ἐπὶ τῶν κατὰ τὴν συνουσίαν ἐντερίου παράτριψις καὶ μετά τινος σπασμοῦ μυξαρίου ἔκκρισις.

Marcus Aurelius

6

Partially monoidal string diagrams

THIS chapter has been adapted from the paper Partially monoidal categories and the algebra of simultaneous substitutions. The paper is joint work with my supervisor Alexander Kurz. This paper wasn't originally published, as it was in some respects supplanted by [45], presented in its extended version in ch. 7.

6.1 Partially monoidal categories

Partial monoids play a role in many different areas of mathematics and computer science. One typical reason for partiality is the one also appearing in resource sensitive logics such as separation logic: If $f : H \to \mathbb{N}$ and $f' : H' \to \mathbb{N}$ are two partial functions from pieces H, H' of the memory, then they can be added if H and H' are disjoint.

In the literature, there are slightly different notions of a partial monoid, depending on the role of the neutral element. A partial monoid can have no neutral element, one neutral element, or many neutral elements (for an example of this, see the definition of a grupoid, first introduced in [46]).

In the following definition, we write $\doteq$ to say that both sides are equal if either side is defined (hence, one side is defined if and only if the other side is).

Definition 6.1. A partial semigroup $(A, \otimes, D)$ consists of a binary operation $\otimes$, defined on $D \subseteq A \times A$ such that for all $a, b, c \in A$ the following

$$(a \otimes b) \otimes c \doteq a \otimes (b \otimes c)$$

A partial monoid $(A, e, \otimes, D)$ has, moreover, a constant e for which

$$e \otimes a \doteq a$$

$$a \otimes e \doteq a$$

These structures are called commutative if $a \otimes b \doteq b \otimes a$.

As explained in the introduction, we are interested in partially monoidal categories. As our examples in this paper are strict, we can give the following simplified definition.

Definition 6.2. A (strict) partially monoidal category, or **p-monoidal** category, consists of

- a category $A = (A_0, A_1)$ with sets A_0 of objects and A_1 of arrows and
- partial monoids $(A_0, e, \otimes, D_0)$ and $(A_1, \mathrm{id}_e, \otimes, D_1)$
- such that $D = (D_0, D_1)$ is a subcategory of $A \times A$
- and $\otimes$ is a functor $D \to A$.

The category is called commutative **p-monoidal** if the two partial monoids are commutative. A strict partially monoidal functor is a functor F such that $F(e) = e$ and $F(a \otimes a') = F(a) \otimes F(a')$ whenever $a \otimes a'$ is defined. The **p-monoidal** categories along with **p-monoidal** functors themselves form a category.

Remark 6.3. In the examples of this paper, the third bullet point could be strengthened to say that D is a full subcategory, that is, two arrows can be composed by $\otimes$ whenever their domains and codomains can be composed.

The fourth bullet point entails the interchange law

$$(f_1 \otimes f_2)\#(g_1 \otimes g_2) \doteq (f_1\#g_1) \otimes (f_2\#g_2) \tag{6.1}$$

whenever $(f_1, f_2) \in D$ and $(g_1, g_2) \in D$.

Here, in the partially monoidal situation, the right-hand side may be defined without the left-hand side being defined. In particular, it will not always be possible to 'slice up' a string diagram in the familiar fashion, see the slashed red line in eq. 6.3 or eq. 6.4 for examples.

We write $\doteq$ to emphasise that this interchange law is weaker than the one for 2-categories, which holds whenever either one of the two sides is defined, see Mac Lane [47](Ch.XII.5).

Similarly to Mac Lane [47], we also give a one-sorted formulation of partially monoidal categories.

Proposition 6.4. *The data of a partially monoidal category can also be described as a category* $(C, s, t, \#)$ *in the sense of (1-4) of [47](Ch.XII.5, p.297) equipped with a partial monoid* $(C, \varepsilon, \otimes, D)$ *where* D *restricts to a subcategory of* $(C, s, t, \#)$ *satisfying the equations*

$$s(c \otimes c') \doteq s(c) \otimes s(c')$$
$$t(c \otimes c') \doteq t(c) \otimes t(c')$$

and the interchange law (eq. 6.1).

Proof. Given the data of the proposition, we reconstruct the data from def. 6.2 as follows. Let $D_1 = D$ be the domain of definition of $\otimes$. Define $A_0 = \{s(f) \mid f \in C\} = \{t(f) \mid f \in C\}$ and $A_1 = C$. All of $\varepsilon, s(\varepsilon), t(\varepsilon)$ are identities on A_0 w.r.t. $\otimes$, hence we can define $e = \varepsilon = s(\varepsilon) = t(\varepsilon)$ to obtain the partial monoid $(A_0, e, \otimes, D_0)$ with D_0 being the restriction of D_1 to A_0. It also follows that $\varepsilon = \mathrm{id}_e$, hence $(A_1, \mathrm{id}_e, \otimes, D_1)$ is the other monoid. And $\otimes$ is a functor since it preserves identities by definition and preserves composition due to the interchange law.

□

A 2-category (C_0, C_1, C_2) almost becomes a **p-monoidal** category by taking the arrows C_1 as the objects A_0, but not quite, since a **p-monoidal** category needs to have a neutral element e. But there is a notion of **p-monoidal** category without unit that comprises 2-categories as a special case.

Example 6.5. Below we give two examples of **p-monoidal** categories:

- We fix a countably infinite set $\mathcal{n}$. The category $\mathrm{n}\mathbb{F}$ of finite subsets of $\mathcal{n}$ with $\oplus$ the partially defined union of disjoint sets is a symmetric **p-monoidal** category. Note that by the union of disjoint sets, we do **not** mean the disjoint/tagged union of sets. We simply mean that $\oplus$ a partial set union operation on only those sets which are disjoint; e.g. $\{1, 2\} \oplus \{2, 3\}$ is undefined, since the intersection $\{1, 2\} \cap \{2, 3\}$ is not empty. On the other hand, we have

$$\{1, 2\} \oplus \{3, 4\} = \{1, 2\} \cup \{3, 4\} = \{1, 2, 3, 4\}$$

We denote by $\underset{\sim}{n\mathbb{B}}$ the subcategory of bijections.

- Another example is the notion of **heaplets**[1] in separation logics. A heaplet is a partial function $\eta : X \rightharpoonup H$ from the address space to data, representing a computer heap. The composition of heaplets is partial operation, defined as $\eta_1 \oplus \eta_2 = \eta_1 \cup \eta_2$, whenever $\mathbf{dom}\ \eta_1 \cap \mathbf{dom}\ \eta_2 = \emptyset$.
  We can then define a subheap relation $\eta \preccurlyeq \gamma$, defined if and only if there exists a heaplet x, s.t. $\eta \oplus x = \gamma$ ($\eta \oplus x$ must of course be defined). Heaplets together with $\preccurlyeq$ form a **p-monoidal** category.

Remark 6.6. [Equivalence of $\underset{\sim}{n\mathbb{F}}$ and $\underset{\sim}{\mathbb{F}}$]

$\underset{\sim}{\mathbb{F}}$ is a category with natural numbers as objects together with all bijective functions between $\underline{n}, \underline{m}$ for every object n, m, where $\underline{n} = \{0, \dots, n-1\}$.

The category $\underset{\sim}{n\mathbb{F}}$ is equivalent as a category to the the skeleton category $\underset{\sim}{\mathbb{F}}$. However, they are not equivalent as partially monoidal categories.

Indeed, $\underset{\sim}{n\mathbb{F}}$ is commutative, but $\underset{\sim}{\mathbb{F}}$ is not. Even though $n+m$ equals $m+n$, the symmetry $\underline{n+m} \to \underline{m+n}$ is not the identity. This is another sense in which the **p-monoidal** category $\underset{\sim}{n\mathbb{F}}$ is easier to work with than the monoidal category $\underset{\sim}{\mathbb{F}}$.

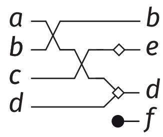
We will see variations on ex. 6.5 in the next section. Semantically, we will have the opportunity to replace sets by words or multisets. Syntactically, we will represent $\underset{\sim}{n\mathbb{B}}$ and $\underset{\sim}{n\mathbb{F}}$ by a string diagrammatic calculus.

In our examples, the monoid operation is the partial union of disjoint sets. There are various ways in which one can turn this operation into a total operation, but that would introduce technicalities that would take us further away from the aim of this paper: Syntactic representations of $\underset{\sim}{n\mathbb{B}}$ and $\underset{\sim}{n\mathbb{F}}$ up to isomorphism that correspond closely to how we work with simultaneous substitutions in an informal way. As emphasised above, we are interested in a mechanism that reflects directly that $[a{\mapsto}b, a{\mapsto}c]$ is not a valid simultaneous substitution.

6.2 Syntax and Semantics

The syntax we will develop in this section is that of nominal string diagrams, such as the following one:

[1] See the slides *Introduction to Separation Logic* at: https://staffwww.dcs.shef.ac.uk/people/G.Struth/mgs18/sl-lec1.pdf



(6.2)

with wires labelled from an infinite, countable alphabet $\mathcal{N}$, the elements of which are written a, b, c etc. The semantics we are interested in is that of functions between finite sets. For example, the diagram above will correspond to the function

$$\{a, b, c, d\} \to \{b, d, e, f\}$$
$$a, d \mapsto d \quad b \mapsto b \quad c \mapsto e$$

There are three different ways to formalise this.

First, we can treat wires as ordered and labelled with elements of $\mathcal{N}$. Sequential composition respects the order of the wires. Parallel composition is partial because distinct wires should be labelled with distinct names. For example, $[a{\mapsto}b] \oplus [a{\mapsto}c]$ is not defined.

Second, we can treat wires as ordered and number them explicitly. The label of a wire is then an occurrence of a, that is, a pair (i, a) where i is a number and a a name. Parallel composition can then be total and distinct wires will still have distinct labels as multiple occurrences of the same name are now distinguished by different indices i. But we need to be careful because we can now build diagrams such as $[a{\mapsto}b, a{\mapsto}c]$ that do not denote functions between subsets of $\mathcal{N}$.

Third, we can treat wires as unordered. Instead of thinking of ordered wires lined up in a linear fashion top to bottom, we now picture them as coming out of plane with no particular order between them. Sequential composition is still uniquely defined as each wire carries a unique label. Another way to look at it is that the rewrite rules of diagrams need to be understood modulo exchange of wires. Accordingly, proofs are one step further away from what would be implemented in a proof assistant but easier for human consumption and closer to geometric intuition.

Each of the three approaches can be understood as dealing in different ways with the simple fact that a set is a list modulo exchange and contraction[2]. In the first and third approach, contraction is built into the data structure by restricting our categories to **irredundant** diagrams. By **irredundant** diagrams, we mean that the inputs/outputs contain no duplicate names. Consequently, parallel composition must be partial. Moreover, in the third approach, exchange is also built in. In the second approach, parallel composition is total and it is possible to build diagrams that are not **irredundant** and do not correspond to functions between sets of names. We show that the first and second approach generate the

[2]removing duplicates

same equivalence relation on repetition-free diagrams. Alternatively, one could investigate adding explicit equations for contractions, which we do not do.

In the following we will discuss these three approaches in turn. We start with some notational preliminaries.

A word of length n is a function $\vec{X} : n \to \mathcal{N}$, that is, an element of $\mathcal{N}^n$. We may also write a word $\vec{X}$ as $(x_0, \dots x_{n-1})$. If we want to emphasise that the range of a word is a set X of names, we write a word as $\vec{X} : n \to X$, or also as $\vec{X} : \|\vec{X}\| \to X$, where $n = \|\vec{X}\|$ denotes the length of the word $\vec{X}$.

Since we are interested in string diagrams representing functions between sets, we sometimes want to restrict attention to words that do not have multiple occurrences of any letter. We call these words **irredundant** and, as mentioned earlier, call a diagram **irredundant** when distinct input/output wires are labelled with distinct names. For example, diagram 6.2 is **irredundant**.

Definition 6.7. A word $\vec{X} : \|\vec{X}\| \to X$ is called **irredundant** if $\vec{X}$ is a bijection.

Writing $|-|$ for the cardinality of a set, if $\vec{X}$ is **irredundant** then $\|\vec{X}\| = |X|$ and we also write $\vec{X} : |X| \to X$.

6.2.1 Ordered sets of wires

Semantically, we consider the category of ordered sets in this section; that is, ordered subsets of names. More formally, we define the category of ordered sets as follows.

Definition 6.8. The category of (finite) ordered sets $\mathsf{sw}\mathbb{F}$ is defined as the category that has **irredundant** words over the alphabet $\mathcal{N}$ as objects and has as arrows $(f, g) : \vec{X} \to \vec{Y}$ commutative squares

$$\begin{tikzcd}
|X| \arrow[r, "f"] \arrow[d, tail, two heads, "\vec{X}"'] & |Y| \arrow[d, tail, two heads, "\vec{Y}"] \\
X \arrow[r, "g"'] & Y
\end{tikzcd}$$

We denote by $\mathsf{sw}\mathbb{B}$ the subcategory where all g (hence f) are bijective[3].

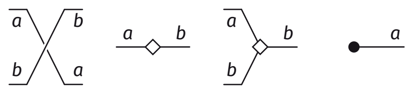
While we are interested in the meaning of a diagram as a function between subsets of $\mathcal{N}$, we start by interpreting them as arrows between words. The reason is that in a diagram the

[3]Since all functions in the commutative square making up an arrow (f, g) are bijective, the functions f and g determine each other.

order of wires matters.

Syntactically, diagrams are arrows in a syntactic category where objects are **irredundant** words and arrows are built up from the basic diagrams and sequential and parallel composition:

Definition 6.9. We write swF for the partially monoidal category that has **irredundant** words as objects and arrows freely generated from instances of



σ δ μ η

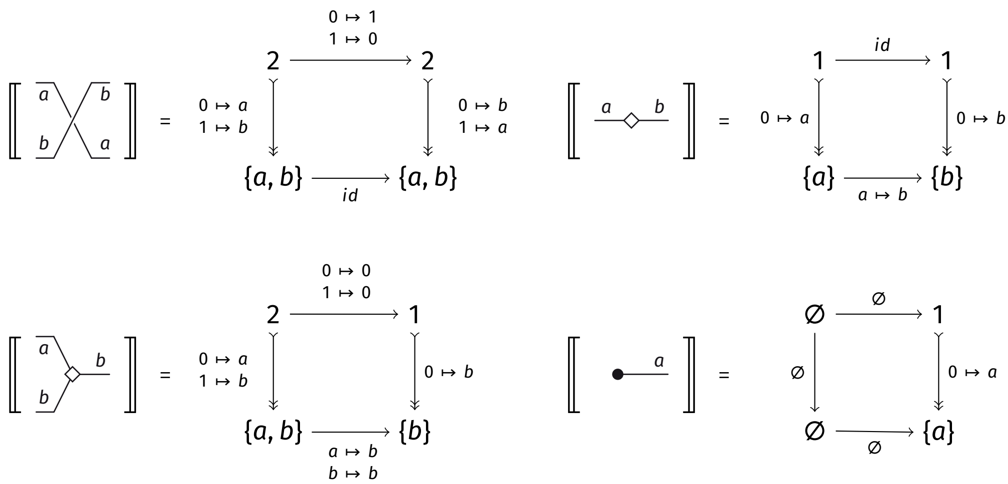
which can be stacked vertically or connected horizontally[4]. The partially monoidal category swB is the subcategory freely generated from instances of σ and δ only.

We now define the interperetation $[\![-]\!]$ of the syntactic objects of swF/swB in sw$\mathbb{F}$/sw$\mathbb{B}$:

Definition 6.10. The basic diagrams

σ (twist), δ (renaming), μ (substitution) and η (lollipop)

are parameterised by distinct $a, b \in \mathcal{N}$ and have the following interpretation as arrows $\mathcal{N}^n \to \mathcal{N}^m$



Next, we generate a partially monoidal category from the above basic diagrams and sequential and parallel composition. Sequential composition of diagrams is given by sequential composition of functions:

[4]Provided the interfaces of the two diagrams match.

$$
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[d, "j"] \\
X \arrow[r, "w"'] & Y
\end{tikzcd}
\; ; \;
\begin{tikzcd}
n \arrow[r, "g"] \arrow[d, "j"'] & o \arrow[d, "k"] \\
Y \arrow[r, "v"'] & Z
\end{tikzcd}
=
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[r, "g"] \arrow[d, "j"] & o \arrow[d, "k"] \\
X \arrow[r, "w"'] & Y \arrow[r, "v"'] & Z
\end{tikzcd}
$$

The notation above implies that the wires that are composed, denoted j, agree in number, order and labelling.

Parallel composition is partial, as it is only defined when $X \cap W = Y \cap V = \varnothing$:

$$
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[d, "j"] \\
X \arrow[r, "w"'] & Y
\end{tikzcd}
\oplus
\begin{tikzcd}
o \arrow[r, "g"] \arrow[d, "k"'] & p \arrow[d, "l"] \\
W \arrow[r, "v"'] & V
\end{tikzcd}
=
\begin{tikzcd}
m + o \arrow[r, "f \oplus g"] \arrow[d, "i \oplus k"'] & n + p \arrow[d, "j \oplus l"] \\
X \uplus W \arrow[r, "w \uplus v"'] & Y \uplus V
\end{tikzcd}
$$

where $\uplus = \cup$, due to the partiality constraint, and $\oplus$ is defined as:

$$
\begin{aligned}
&f \oplus g : m + o \to n + p, \\
&f \oplus g(j) = \begin{cases} f(j) & \text{for } 0 \leq j < m \\ g(j - m) + p & \text{for } m \leq j < m + o \end{cases}
\end{aligned}
$$

and

$$
\begin{aligned}
&i \oplus k : m + o \to X \uplus W, \\
&i \oplus k(j) = \begin{cases} i(j) & \text{for } 0 \leq j < m \\ k(j - m) & \text{for } m \leq j < m + o \end{cases}
\end{aligned}
$$

Finally, recall that $\mathsf{n}\mathbb{F}$ and $\mathsf{n}\mathbb{B}$ denote the partially monoidal categories of (finite) functions and bijections, respectively. We now have

$$\mathsf{swF} \xrightarrow{[\![-]\!]} \mathsf{sw}\mathbb{F} \xrightarrow{|-|} \mathsf{n}\mathbb{F}$$

and

$$\mathsf{swB} \xrightarrow{[\![-]\!]} \mathsf{sw}\mathbb{B} \xrightarrow{|-|} \mathsf{n}\mathbb{B}$$

where the forgetful functor $| - |$ maps an arrow (f, g) of words to the function g, such that the following holds.

> **Proposition 6.11.** *The semantics extends to partially monoidal functors* $|[\![-]\!]| : \mathsf{swB} \to \mathsf{n}\mathbb{B}$ *and* $|[\![-]\!]| : \mathsf{swF} \to \mathsf{n}\mathbb{F}$. *In particular, if* $(f, g) = [\![\phi : w \to v]\!]$, *then* g *is a function from the set of letters of* w *to the set of letters of* v. *Moreover, if* ϕ *is in* swB *then* g *is a bijection.*

In sec. 6.3 we are going to axiomatise the theory of bijections that describes which diagrams are identified by $|[\![-]\!]| : \mathsf{swB} \to \mathsf{n}\mathbb{B}$ and in sec. 6.4 the theory of functions that describes which diagrams are identified by $|\llbracket - \rrbracket| : \underset{\sim}{\mathsf{swF}} \to \underset{\sim}{\mathsf{n}}\mathbb{F}$.

6.2.2 Ordered multisets of wires

The aim of this section is to investigate what happens if we make parallel composition total. One reason for doing this is that we will prove that even though the resulting rewriting system may take detours via meaningless diagrams, it is the case that every rewrite in the 'total system' between two **irredundant** words corresponds to some rewrite in the 'partial system'.

For example, in this section we will allow the composition $[a \mapsto b] \oplus [a \mapsto c] = [a \mapsto b, a \mapsto c]$. Semantically, we make this correspond to a function $\{(0, a) \mapsto (0, b), (1, a) \mapsto (1, c)\}$ not between sets but occurrences of names. Accordingly, in the semantics, we will use words of pairs $((0, x_0), \ldots (n-1, x_{n-1}))$ instead of words $(x_0, \ldots x_{n-1})$.

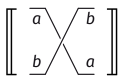
Technically, going to a total parallel composition corresponds to going from ordered sets of names to ordered sets of occurrences of names, or from ordered sets to ordered multisets (ordered multisets are pomsets [48] where the order happens to be linear), or from **irredundant** words to words.

Definition 6.12. The category of words with functions $\underset{\sim}{\mathsf{w}}\mathbb{F}$ is defined as the category of words over the alphabet $\mathcal{N}$ with an arrow $(f, g) : \vec{X} \to \vec{Y}$ being a commutative square,

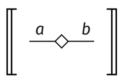
$$\begin{tikzcd} \|\vec{X}\| \arrow[r, "f"] \arrow[d, tail, "\vec{X}"'] & \|\vec{Y}\| \arrow[d, tail, "\vec{Y}"] \\ X \arrow[r, "g"'] & Y \end{tikzcd}$$

modulo an equivalence relation on arrows $(f, g) \simeq (f', g')$ if $f = f'$. We denote by $\underset{\sim}{\mathsf{w}}\mathbb{B}$ the subcategory of arrows (f, g) where f is bijective.

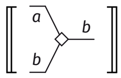
The equivalence relation on arrows is justified by the observation that, on the image of $\vec{X}$, the arrow g is determined by f. (The reason we are only interested in the image of $\vec{X}$ is that this image determines the word uniquely.)



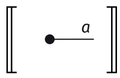
$$
= \quad \begin{tikzcd}
2 \arrow[r, "\substack{0\mapsto 1\\ 1\mapsto 0}"] \arrow[d, "\substack{0\mapsto(0,a)\\ 1\mapsto(1,b)}"'] & 2 \arrow[d, "\substack{0\mapsto(0,b)\\ 1\mapsto(1,a)}"] \\
2\times\{a,b\} \arrow[r, "\substack{(0,a)\mapsto(1,a)\\ (1,b)\mapsto(0,b)}"'] & 2\times\{a,b\}
\end{tikzcd}
$$



$$
= \quad \begin{tikzcd}
1 \arrow[r, "id"] \arrow[d, "0\mapsto(0,a)"'] & 1 \arrow[d, "0\mapsto(0,b)"] \\
1\times\{a\} \arrow[r, "(0,a)\mapsto(0,b)"'] & 1\times\{b\}
\end{tikzcd}
$$



$$
= \quad \begin{tikzcd}
2 \arrow[r, "\substack{0\mapsto 0\\ 1\mapsto 0}"] \arrow[d, "\substack{0\mapsto(0,a)\\ 1\mapsto(1,b)}"'] & 1 \arrow[d, "0\mapsto(0,b)"] \\
2\times\{a,b\} \arrow[r, "\substack{(0,a)\mapsto(0,b)\\ (1,b)\mapsto(0,b)}"'] & 1\times\{b\}
\end{tikzcd}
$$



$$
= \quad \begin{tikzcd}
\varnothing \arrow[r, "\varnothing"] \arrow[d, "\varnothing"'] & 1 \arrow[d, "0\mapsto(0,a)"] \\
\varnothing \arrow[r, "\varnothing"'] & 1\times\{a\}
\end{tikzcd}
$$

Next, we generate a partially monoidal category from the above basic diagrams by closing under sequential and parallel composition. Sequential composition of diagrams is given by sequential composition of functions:

$$
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[d, "j"] \\
m\times X \arrow[r, "w"'] & n\times Y
\end{tikzcd}
\; ; \;
\begin{tikzcd}
n \arrow[r, "g"] \arrow[d, "j"'] & o \arrow[d, "k"] \\
n\times Y \arrow[r, "v"'] & o\times Z
\end{tikzcd}
=
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[r, "g"] \arrow[d, "j"] & o \arrow[d, "k"] \\
m\times X \arrow[r, "w"'] & n\times Y \arrow[r, "v"'] & o\times Z
\end{tikzcd}
$$

The parallel composition is now total:

$$
\begin{tikzcd}
m \arrow[r, "f"] \arrow[d, "i"'] & n \arrow[d, "j"] \\
m\times X \arrow[r, "w"'] & n\times Y
\end{tikzcd}
\;\oplus\;
\begin{tikzcd}
o \arrow[r, "g"] \arrow[d, "k"'] & p \arrow[d, "l"] \\
o\times W \arrow[r, "v"'] & p\times V
\end{tikzcd}
=
\begin{tikzcd}
m+o \arrow[r, "f\oplus g"] \arrow[d, "i\oplus k"'] & n+p \arrow[d, "j\oplus l"] \\
(m+o)\times(X\cup W) \arrow[r] & (n+p)\times(Y\cup V)
\end{tikzcd}
$$

Definition 6.13. We write wF for the monoidal category that has words as objects and has arrows that are freely generated from instances of $\sigma, \delta, \mu, \eta$. wB is the monoidal subcategory generated from σ and δ only.

Proposition 6.14. *The semantics extends to monoidal functors* $[\![-]\!] : \mathrm{wB} \to \mathrm{w}\mathbb{B}$ *and* $[\![-]\!] : \mathrm{wF} \to \mathrm{w}\mathbb{F}$.

The next proposition says that if a diagram in wF is **irredundant**, then it induces, and is determined by, a unique function between sets of names (denoted g' in the proposition).

Proposition 6.15. *Let* $\phi : \vec{X} \to \vec{Y}$ *be a diagram in* wF *and* $(f, g) = [\![\phi]\!]$. *If* $\vec{X}$ *and* $\vec{Y}$ *are* ***irredundant****, then* $g : \|\vec{X}\| \times X \to \|\vec{Y}\| \times Y$ *is of the form* $g = f \times g'$ *for a unique function* $g' : X \to Y$.

Proof. This follows since **irredundant**ness means that $\vec{X}$ and $\vec{Y}$ are bijections. In detail, we have

$$\begin{tikzcd} {|X|} \arrow[r, "f"] \arrow[d, "\langle \mathrm{id}, \vec{X} \rangle"'] & {|Y|} \arrow[d, "\langle \mathrm{id}, \vec{Y} \rangle"] \\ {|X| \times X} \arrow[r, "g"'] & {|Y| \times Y} \end{tikzcd}$$

and define $g' = \vec{Y} \circ f \circ \vec{X}^{-1}$.

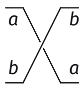
The equation $(f \times g') \circ \langle \mathrm{id}, \vec{X} \rangle = \langle \mathrm{id}, \vec{Y} \rangle \circ f$ follows immediately as well as that any g' satisfying this equation is uniquely determined.

□

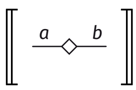
6.2.3 Sets of wires

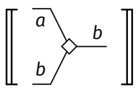
In this section, we change the notion of sequential composition so that it ignores the ordering of the wires. This is possible because, as in sec. 6.2.1, every wire will carry a unique label. Thus, the domain and codomain of a diagram $\phi : X \to Y$ are sets of wires.

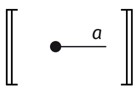
In secs. 6.2.1, 6.2.2, even though the generator

$\sigma =$ 

defines the identity function $\{a, b\} \to \{a, b\}$, we could not add the equation $\sigma = \mathrm{id}$ as sequential composition had to respect the order of the wires and the labels. With the new sequential composition we could add this equation, but it seems easier to just drop σ from the generators and to take the domain and codomain of a diagram to be sets of labels rather than words. The semantics of δ (renaming), μ (substitution), and η (lollipop) can then be given directly in terms of functions.

 $= \quad \{a\} \xrightarrow[a \mapsto b]{} \{b\}$

 $= \quad \{a, b\} \xrightarrow[\substack{a \mapsto b \\ b \mapsto b}]{} \{b\}$

 $= \quad \varnothing \xrightarrow[\varnothing]{} \{a\}$

Sequential composition of diagrams is described as above by linking wires with the same label. Parallel composition is stacking diagrams on top of each other and is partial as it has to respect the **irredundant**ness constraints.

Definition 6.16. We write nF for the partially monoidal category freely generated from δ, μ, η and parallel and modified sequential composition as described above. We write nB for the partially monoidal category freely generated from δ only.

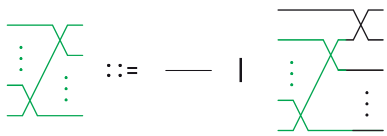
Proposition 6.17. *The semantics extends to partially monoidal functors* $[\![-]\!] : \underset{\sim}{\mathrm{nB}} \to \underset{\sim}{\mathrm{n}\mathbb{B}}$ *and* $[\![-]\!] : \underset{\sim}{\mathrm{nF}} \to \underset{\sim}{\mathrm{n}\mathbb{F}}$*.*

6.3 The Theory of Bijective Functions

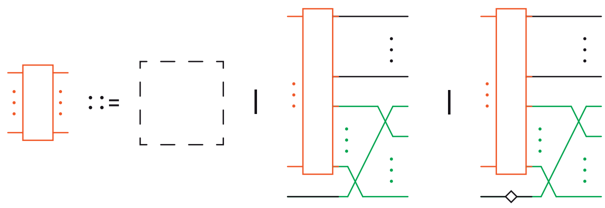
In this section we will consider the category $\underset{\sim}{\mathrm{n}\mathbb{B}}$ of bijections of finite subsets of some set $\mathcal{N}$, with the generator $\delta_{ab} : \{a\}^1 \to \{b\}^1$ representing a bijection $\{a\} \rightarrowtail\!\!\!\!\twoheadrightarrow \{b\}$ and $\sigma_{ab} : \{a, b\}^2 \to \{a, b\}^2$ representing the identity function $id_{\{a,b\}} : \{a, b\} \to \{a, b\}$.

Note. We write $\{a, b\}^2$ to mean a function $2 \to \{a, b\}$.

Following Lafont [49], we introduce a notion of a canonical form for string diagrams formed from the generators σ and δ, defined in the previous section. We first inductively define the notion of *stairs*:



Next we define the *canonical form*:



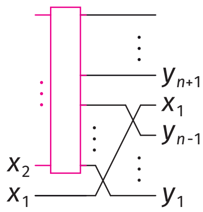
In both instances, we omit the names/labels on the wires for better readability.

Lemma 6.18. *Any bijective function* $f : X \rightarrowtail\!\!\!\!\twoheadrightarrow Y$ *together with an ordering on* X *and* Y *(given by* $\vec{X} : |X| \rightarrowtail\!\!\!\!\twoheadrightarrow X$ *and* $\vec{Y} : |Y| \rightarrowtail\!\!\!\!\twoheadrightarrow Y$*) is represented by a unique canonical form*[5].

Proof. By induction on the size n of X and Y:

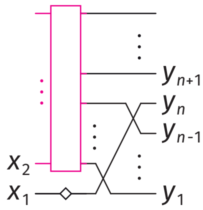
- If $n = 0$, then f is the identity function on the empty set and is represented by the empty string diagram.
- If $n \geq 1$, then given $\vec{X}$ and $\vec{Y}$, we have $x_1 = \vec{X}(1)$ and $y_n = f(x_1)$ (where $n = \vec{Y}^{-1}(f(x_1))$). Now, we have two cases, either $x_1 = y_n$, in which case we will have the diagram:

[5] The canonical diagrams are unique in the graphical 2D syntax and unique up to axioms of a monoidal category when represented in the one dimensional syntax.



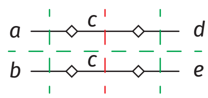
where the remaining part (in magenta) is given by the IH, by removing x_1 and y_n from $f, \vec{X}$ and $\vec{Y}$ and re-numbering the order functions.

In case $x_1 \neq y_n$, we get the following diagram (with the IH giving the remaining part, as in the previous case):



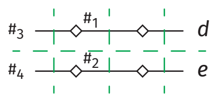
□

Before we show that the rewriting system of fig. 6.1 is terminating and rewrites to the canonical form, we have to take care of the fact that due to the partiality of $\oplus$ not all diagrams can be decomposed in the usual fashion. Consider the example below:



(6.3)

Here, the vertical slicing of the diagram in the middle is not allowed, because the two sub-diagrams would violate the **irredundant**ness constraint, since two c's would appear in the codomain and domain of the left and right sub-diagram, respectively. In order to avoid such conflicts, we define a notion of restricted substitution, wherein we replace repeated names with fresh ones, such that the new diagram can be sliced arbitrarily without restriction. For example, the substitution renames both c's into fresh variables:



As we will show, the rewriting system in fig. 6.1 axiomatises the theory of bijections. Notice that we elided the labels in some of the equations, which then need to be instantiated by appropriately labelling the wires as shown in sec. 6.2.1.

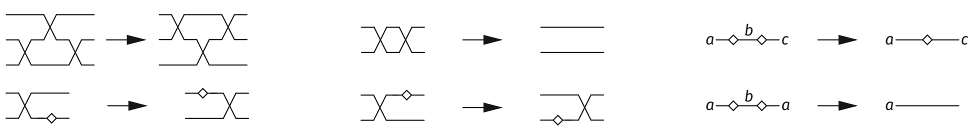


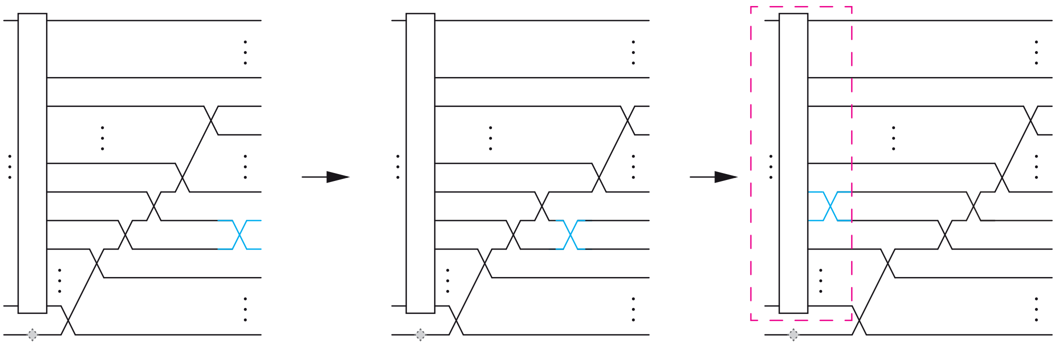
Figure 6.1: Rewrite rules of swB

Lemma 6.19. *Any diagram $\phi : \vec{X} \to \vec{Y}$ in which all internal names are fresh reduces to a canonical form $\widehat{\phi}$ by the rules of fig. 6.1.*

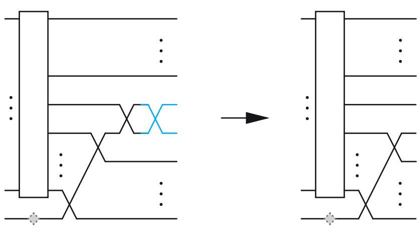
Proof. We prove this lemma by double induction on the number p of input/output ports ($p = |\vec{X}| = |\vec{Y}|$) together with the *size* s of ϕ, defined as the number of generators which make up the diagram.

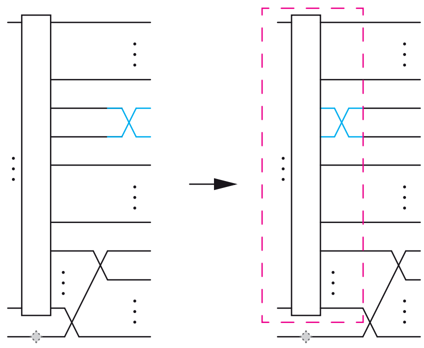
- If $s = 0$, then $\phi = id_{\vec{X}} = id_{\vec{Y}}$, which is a canonical form.
- If $s \geq 1$ then we would like to separate the diagram into an elementary diagram ε and a diagram ψ of size $s - 1$, s.t. $\phi = \psi \,;\, \varepsilon$. However, we need to be careful as this might not always be possible.
  - If $\varepsilon = \sigma$, the proof mirrors Lafont's in the usual way. There are 4 cases (the case for the canonical form with and without the diamond are exactly the same in these 4 cases and we highlight ε in cyan):

    In the first case, we apply the first rule and then apply the IH to a sub-diagram with $p - 1$ ports (outlined in magenta).

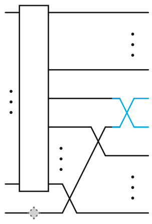


    In the second case, we apply the second rule and obtain a diagram in canonical form. In the third case, we use the IH for $p - 1$ again.



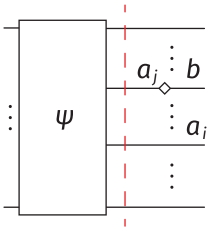


    The last case is a canonical form already.



- If $\varepsilon = \delta$, things are a bit more tricky as we cannot always decompose the diagram in the desired way.

  Consider the following case:

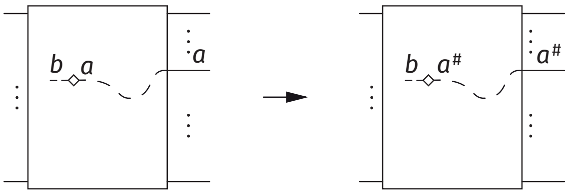


(6.4)

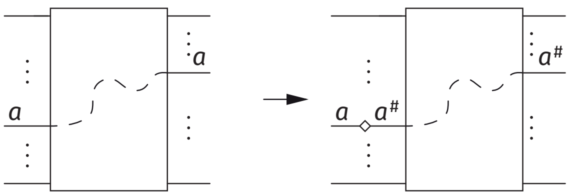
  The diagram above is problematic, as we we cannot split it along the dashed line, since the smaller diagram ψ will no longer be **irredundant**.

  In order to proceed, we use the notion of restricted substitution described earlier, replacing one specific occurrence of a label with a fresh name, starting from the output port going backwards. There will be two cases:

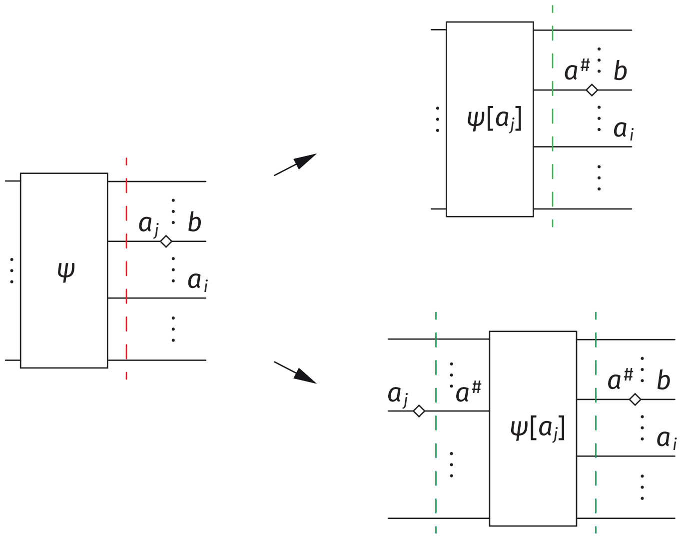
  In the first case, the operation traces the label a backwards, replacing it with a fresh label $a^\#$, until it reaches a diamond:



  In the other case, the label is traced all the way back to the input port, in which case we get the following diagram:



  Applying this operation to the problematic diagram we obtain two cases:

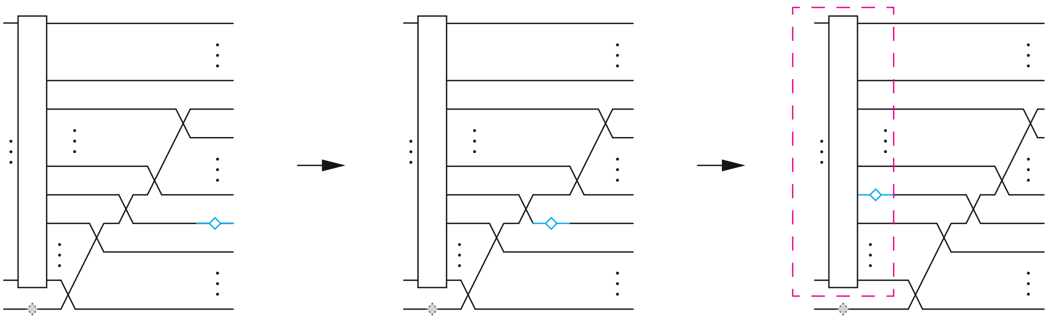


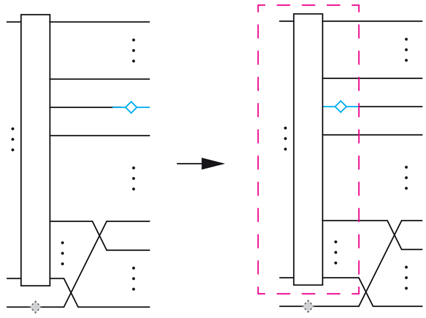
This now allows us to separate the original diagram ϕ into $\psi[a_j]$; $(\vec{id} \oplus \delta_{a^\# b} \oplus \vec{id})$ or $(\vec{id} \oplus \delta_{a_j a^\#} \oplus \vec{id})$; $\psi[a_j]$; $(\vec{id} \oplus \delta_{a^\# b} \oplus \vec{id})$.

Since, by definition, $a^\#$ is a fresh variable not appearing anywhere in the diagram, this decomposition is defined and since $size(\psi[a_j]) = size(\psi) = s - 1$, we can apply the IH to $\psi[a_j]$ and obtain $\widehat{\psi[a_j]}$, which is in canonical form.

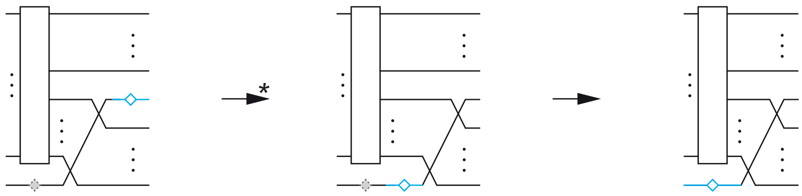
We analyze the following three cases of the sub-diagram $\widehat{\psi[a_j]}$; $(\vec{id} \oplus \delta_{a^\# b} \oplus \vec{id})$, ignoring the second case of the substitution above (for now).

For the first case, we simply slide the diamond past the twist (4th rule) and apply IH to a sub-diagram with $p - 1$ ports (likewise for the second case):

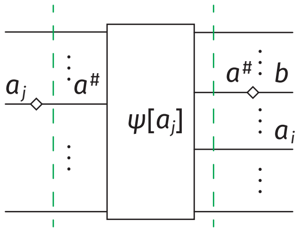




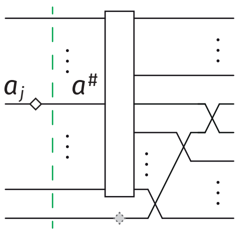
The last case involves multiple application of the last rule, sliding the diamond past all the twists of the *stairs* (this derived rule is easily proven by induction on the height of the *stairs*). Here, we have two further cases. Either the normal form has a diamond at the bottom most port, in which case we apply rule 3 and obtain a diagram in canonical form, otherwise, the last port is an identity and we already have a canonical form.



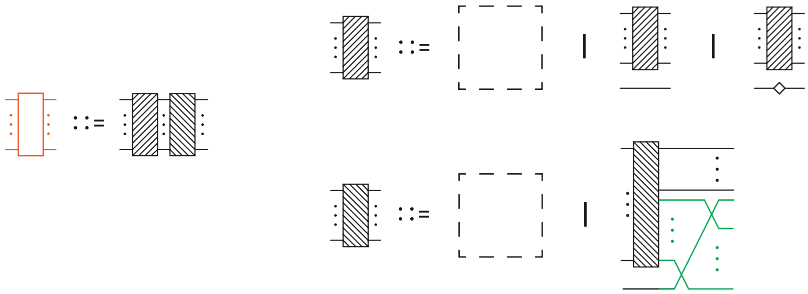
Finally, we get back to the other case of the diagram substitution, namely:



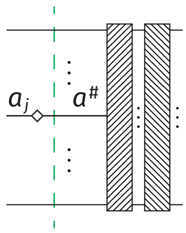
We apply the same reasoning as for the first substitution case and obtain the following diagram (which is almost in canonical form):



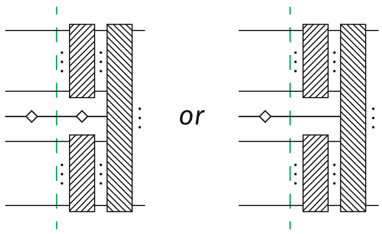
In order to show that the diagram above is/can be turned into canonical form, we will modify the definition of canonical form slightly. One can easily see that the following definition of canonical form is equivalent to the one introduced earlier:



Thus, the previous diagram can be rewritten as:



Which, according to the definition above is the same as:



For the first case, we apply the 3rd rule and obtain canonical form and in the second case, we already have canonical form.

□

Recall from sec. 6.2.1 that the partially monoidal category swB is 'free over twist and diamond' and that there is a **p-monoidal** functor

$$[\![-]\!] : \mathsf{swB} \to \mathsf{sw}\mathbb{B}$$

into the category of **irredundant** words with bijections. We have shown that the kernel of this functor is axiomatised by the equations of fig. 6.1:

Theorem 6.20. *The partially monoidal category swB modulo the equations from fig. 6.1 is isomorphic to the partially monoidal category $\mathsf{sw}\mathbb{B}$ of **irredundant** words with bijections.*

Proof. A quick inspection verifies that the left and right-hand side of all rewrites in fig. 6.1 are mapped to the same bijection between **irredundant** words in $\mathsf{sw}\mathbb{B}$. This shows the soundness of the equations. For completeness, we need to show that if two diagrams ϕ, ψ are identified by $[\![-]\!]$, then they can be proved equal using the equations.

By lem. 6.18, we know that $[\![\phi]\!] = [\![\psi]\!]$ has a unique canonical form and by lem. 6.19 we know that both ϕ and ψ rewrite to this canonical form. Hence ϕ and ψ are equal according to the equations.

$\square$

The next theorem shows that when using equational reasoning, we can work with a total parallel composition. While not all diagrams showing up in such a proof correspond to functions between sets of names, they do so if their domain and codomain are **irredundant** words.

Recall $[\![-]\!] : \mathsf{wB} \to \mathsf{w}\mathbb{B}$ from prop. 6.14.

Theorem 6.21. *Let ϕ, ψ be two* ***irredundant*** *diagrams in wB such that $\phi = \psi$ in the equational theory of wB plus the equations of fig. 6.1. Then $\phi = \psi$ in the equational theory of swB plus the equations of fig. 6.1.*

Proof. There are more equations in wB than in swB because the interchange law in swB is restricted by the partiality of parallel composition, see eq. 6.3. Nevertheless, if $\phi = \psi$ in the equational theory of wB then $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{w}\mathbb{B}$ because the equations of fig. 6.1 are sound wrt to $[\![-]\!] : \mathsf{wB} \to \mathsf{w}\mathbb{B}$. But since ϕ and ψ are **irredundant**, $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{w}\mathbb{B}$ implies $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{sw}\mathbb{B}$. Now the result follows from the completeness part of thm. 6.20.

$\square$

Next, we come to the question of representing the category of $\mathsf{n}\mathbb{B}$ up to isomorphism. Intuitively, the presentation swB plus the equations of fig. 6.1 present $\mathsf{n}\mathbb{B}$ up to isomorphism once we add equations between objects identifying all words that only differ in the order of their letters. This can be made precise by adapting the notion of *presentation modulo* of Curien and Mimram [50] to partially monoidal categories:

The category presented by swB plus the equations of fig. 6.1 plus equational generators [50](Def.7) identifying words that only differ in the order of their letters is isomorphic to the category $\mathsf{n}\mathbb{B}$ of finite sets of names.

If we are willing to work with sets of wires instead of words of wires we obtain the following. Recall prop. 6.17.

Theorem 6.22. *The partially monoidal category nB modulo equations*

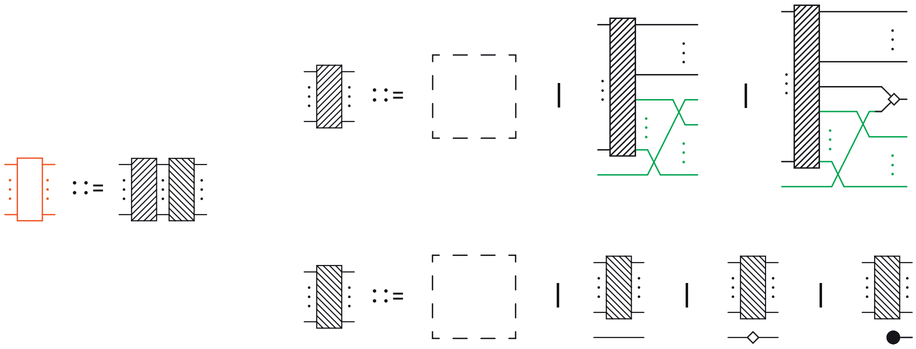
is isomorphic to the partially monoidal category $\mathsf{n}\mathbb{B}$ of finite sets of names with bijections.

Why are the equations of thm. 6.22 so much simpler than the equations of fig. 6.1? Geometrically, if wires are sets, then wires do not line up in one dimension but can be pictured as coming out of a 2-dimensional plane as in fig. 3.1 of [13]. Similarly to how the geometry of planar string diagrams trivialises the laws of monoidal categories, going from ordered wires to sets of wires trivialises the equations of fig. 6.1 that involve twisting of wires.

6.4 The Theory of Functions

We extend the results from the previous section from bijections to functions. In other words, going back to def. 6.9, we extend swB with the generators μ and η, see def. 6.10.

Again, we define a canonical form for these diagrams:



The rules of this rewrite system are given in fig. 6.2.

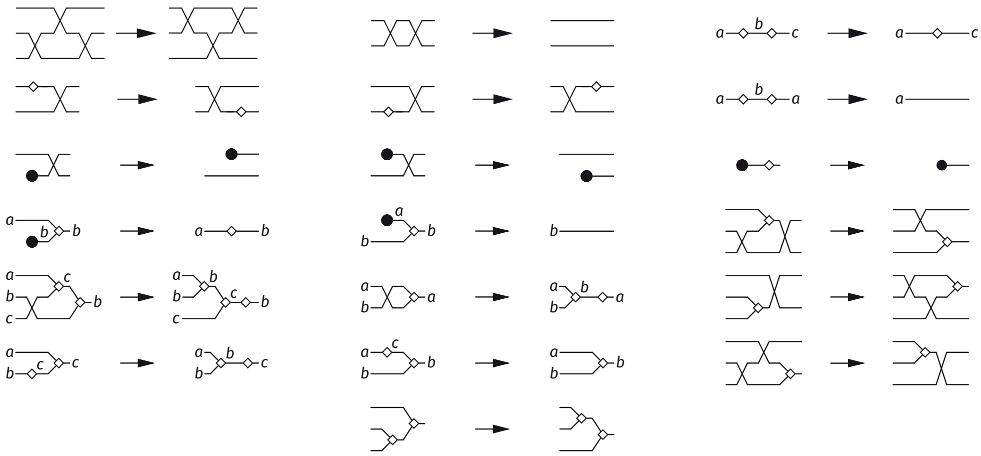


Figure 6.2: Rewrite rules of swF

Lemma 6.23. *Any function $f : \vec{X} \to \vec{Y}$ is represented by a unique canonical form in* swF.

Proof. We begin with the observation that any function $g : X \to Y$ can be factored as a surjection, followed by an injection in a straightforward way. Thus we can do the following factorization for our function f:

$$\begin{tikzcd}
{|X|} \arrow[d, tail, two heads, "\vec{X}"'] & {|X'|} \arrow[d, tail, two heads, "\vec{X'}"] & {|Y|} \arrow[d, tail, two heads, "\vec{Y}"] \\
X \arrow[r, two heads, "f_{sur}"'] \arrow[rr, bend right, "f"'] & X' \arrow[r, tail, "f_{inj}"'] & Y
\end{tikzcd}$$

We will first focus on the left part of the picture, namely the surjection. We define f_{sur} in the following way:

$$f_{sur}(x) = max^{\vec{X}}(2^{f}(\{f(x)\}))$$

where $2^f : 2^Y \to 2^X$ is the pre-image of f and $max^{\vec{X}} : 2^X \to X$ is defined as:

$$max^{\vec{X}}(Z) = \vec{X}(max(\vec{X}^{-1}[Z]))$$

($\vec{X}^{-1}[Z] : 2^X \to 2^{|X|}$ is the inverse function of $\vec{X}$ lifted to sets)

Intuitively, f_{sur} acts as the identity on everything, but the elements, which are identified in the image of f. For those, we take the pre-image of f and choose a canonical/maximal element, which is given to us by the $max^{\vec{X}}$ function. This function takes the set of elements that are to be identified and choses the largest one, according to the given ordering $\vec{X}$.

Next we need to give the ordering function $\vec{X'}$. Since this must be a bijection, we will instead give the definition of the inverse function $\vec{X'}^{-1}$:

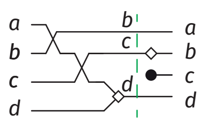
$$\vec{X'}^{-1}(x) = \textbf{shift}\,(\vec{Y}^{-1}(f(x)), \vec{Y}^{-1}(f(x)) + 1)$$

$$\textbf{shift} : |Y| \times |Y| \to |X|$$

$$\textbf{shift}\,(g, 0) = g$$

$$\textbf{shift}\,(g, c) = \begin{cases} \textbf{shift}\,(g, c-1) & \text{if } \vec{Y}(c-1) \in f[X] \\ \textbf{shift}\,(g-1, c-1) & \text{otherwise} \end{cases}$$

The ordering $\vec{X'}$ is defined from the ordering $\vec{Y}$, by essentially composing $\vec{Y}$ with f and then filtering out the elements in the domain of $\vec{Y}$, which do not appear in the image of f. It is much easier to see this pictorially:



In the diagrammatic representation of a function f above, f_{sur} is defined as

$$f_{sur}(a) = d \;\; f_{sur}(b) = b \;\; f_{sur}(c) = c \;\; f_{sur}(d) = d$$

and can be read off from the sub-diagram to the left of the dashed line. The ordering $\vec{X}'$ is:

$$\vec{X}'(0) = b \;\; \vec{X}'(1) = c \;\; \vec{X}'(2) = d$$

and thus

$$\vec{X}'^{-1}(b) = 0 \;\; \vec{X}'^{-1}(c) = 1 \;\; \vec{X}'^{-1}(d) = 2$$

We can verify that our definition of $\vec{X}'^{-1}$ above is correct, by checking that $\vec{X}'^{-1}(d)$ is really 2.

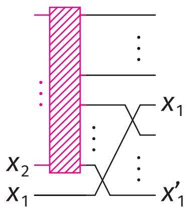
$$\begin{aligned}
\vec{X}'^{-1}(d) &= \textbf{shift}\,(\vec{Y}^{-1}(f(d) + 1), \vec{Y}^{-1}(f(d))) && \\
&= \textbf{shift}\,(\vec{Y}^{-1}(d), \vec{Y}^{-1}(d) + 1) && \\
&= \textbf{shift}\,(3, 4) && \\
&= \textbf{shift}\,(3, 3) && \vec{Y}(3) \in f[X]\ (\vec{Y}(3) = d) \\
&= \textbf{shift}\,(2, 2) && \vec{Y}(2) \notin f[X]\ (\vec{Y}(2) = c) \\
&= \textbf{shift}\,(2, 1) && \vec{Y}(1) \in f[X]\ (\vec{Y}(1) = b) \\
&= \textbf{shift}\,(2, 0) && \vec{Y}(0) \in f[X]\ (\vec{Y}(1) = a) \\
&= 2 &&
\end{aligned}$$

Finally, the right side of the picture, namely the definition of $f_{inj} : X' \to Y$ is simply:

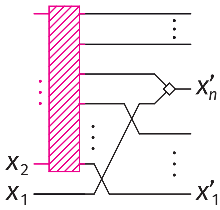
$$f_{inj}(x) = f(x)$$

We will now match the surjective decomposition of f to the left side of the canonical form form and the injective decomposition will correspond to the right canonical diagram.

- Surjection: We proceed by induction on the size n of X:
  - If $n = 0$, then f is the identity function on the empty set and is represented by the empty string diagram.
  - If $n \geq 1$, then given $\vec{X}$ and $\vec{X}'$, we have $x_1 = \vec{X}(1)$ and $x'_n = f_{sur}(x_1)$ (where $n = \vec{X}'^{-1}(f_{sur}(x_1))$). We have two cases. Either x_1 is mapped to x_1 and no other value is identified with x_1 (i.e. $|2^f(\{f(x_1)\})| = 1$), in which case we will have the diagram:

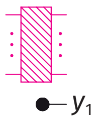


otherwise, we have $|2^f(\{f(x_1)\})| > 1$ and x_1 is identified with some x'_n, in which case we have:

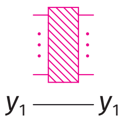


where the rest of the diagram is given by the IH, by re-numbering $\vec{X}, \vec{X}'$ and removing x_1 from the domain of f_{sur}.

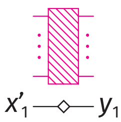
- Injection: Again, we proceed by induction, now on the size n of Y:
  - If $n = 0$, then f is the identity function on the empty set and is represented by the empty string diagram.
  - If $n \geq 1$, then we have $y_1 = \vec{Y}(1)$. In case we have $y_1 \notin f[X]$, we get the diagram:

    Otherwise we have $2^{f_{inj}}(\{y_1\}) = \{x'_1\}$ where we either have $x'_1 = y_1$, in which case we get:

    otherwise:

    The rest of the diagram in all three cases is given by the IH, by re-numbering $\vec{Y}$ and removing x'_1 from the domain of f_{inj}.

□

Lemma 6.24. *The rewrite system in fig. 6.2 is terminating.*

Proof. In order to prove termination, we will use an argument of polynomial interpretation, similar to the one found in Lafont [49].

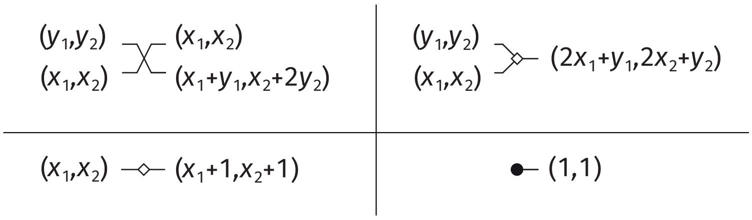
For all diagrams $\sigma : \vec{X} \to \vec{Y}$ we will define a strictly monotonic map $[\sigma] : (N^+ \times N^+)^{|X|} \to (N^+ \times N^+)^{|Y|}$, where N^+ is the set of strictly positive integers and $(N^+ \times N^+)^n$ comes with a lexicographic product order:

$$((x_{11}, x_{12}), \dots, (x_{n1}, x_{n2})) \leq ((y_{11}, y_{12}), \dots, (y_{n1}, y_{n2})) \text{ whenever}$$

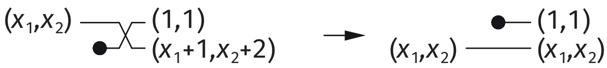
$$(x_{11}, x_{12}) \leq (y_{11}, y_{12}), \dots, (x_{n1}, x_{n2}) \leq (y_{n1}, y_{n2})$$

where $(x_1, x_2) \leq (y_1, y_2) \overset{\text{def}}{=} x_1 \leq y_1 \vee (x_1 = y_1 \wedge x_2 \leq y_2)$

We give a pair of interpretation functions for each of the generators:



This interpretation is compatible with the parallel and sequential composition, and it thus suffices to check that all the rewrite rules, when interpreted, strictly decrease in at least one coordinate. In the example below, this condition is satisfied, since $(x_1, x_2) < (x_1 + 1, x_2 + 2)$:



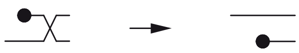
We will omit the rest of the rules. The full proof was formalized and checked using an SMT solver and is discussed in further detail in sec. 6.5.

□

Lemma 6.25. *The canonical form, defined at the beginning of this section is a normal form for the rewriting system, presented in fig. 6.2.*

Proof. To see why the canonical form is a normal form, we analyze the rules of the system and argue that the canonical form must be the normal form because it contains no redexes.

Looking at the canonical form, we can see that it is split into the a left and a right canonical form, where the left diagram contains only twists and cups and the right side only contains diamonds and lollipops. We can thus eliminate all the rules which have a lollipop/diamond before a cup or a twist, such as

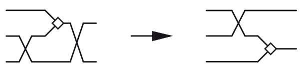


since the diagram on the left of such rule can appear neither in the left nor in the right canonical diagram.

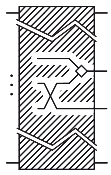
For rules, such as

the left hand sides clearly cannot appear in the left canonical form, and by inspecting the right canonical form, we can see that they also cannot appear there, as the canonical right diagram can only ever have one generator at any given port/level.

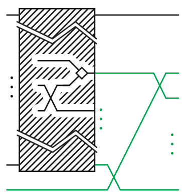
A similar argument can be made for the rest of the rules, which involve analyzing the left normal form diagram. Take for example



We need to show that the left hand side of the rule could never appear in a canonical form. Due to the shape of the rule we must necessarily have the following diagram:



It is then easy to see that there is no way to attach the second twist, such that the resulting diagram is a canonical one, since the only way to attach a twist this diagram is to use stairs, which will lead to

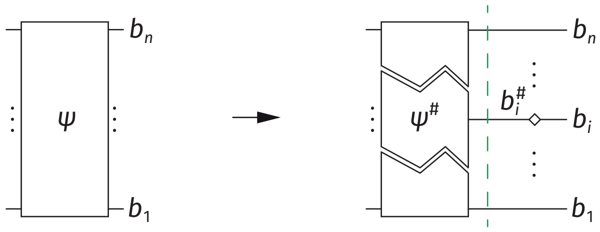


The other rules follow in a similar fashion.

□

Lemma 6.26. *The rewrite system in fig. 6.2 is locally confluent and reduces to the canonical form.*

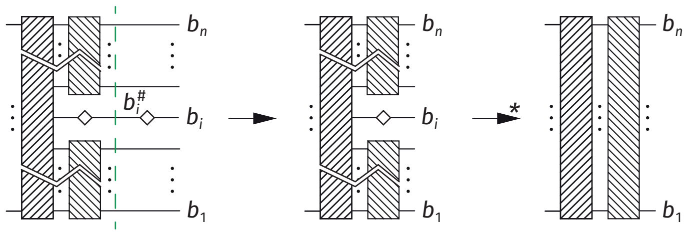
Proof. To avoid the difficulty of the partiality of the parallel composition, we will use a similar trick as in lem. 6.19, wherein we substitute multiple occurrences of the same variable with fresh ones, s.t. we get the following:



Since the diagram $\psi^\#$ (before the dashed line) only contains one occurrence of each name, all compositions are defined (informally, this is because any sub-diagram will have a **irredundant** word on both the input and output ports). It now suffices to show that $\psi^\#$ reduces to normal form, by proving local confluence.

To show local confluence, ~100 critical pairs have to be checked[6]. These are omitted for brevity and can be found on github[7].

Having shown that the normal form is indeed the canonical form in lem. 6.25, the reduced diagram $\widehat{\psi^\#}$ is of the form:



In order to get the final canonical from, we simply need to apply the 3rd or 6th rule to collapse the two diamonds for all the diamonds introduced by the substitution operation.

□

The remainder of this section copies almost verbatim the corresponding part in sec. 6.3. Recall from sec. 6.2.1 that the partially monoidal category swF is 'free over twist, diamond, cup and lollipop' and that there is a functor

$$[\![-]\!] : \text{swF} \to \text{sw}\mathbb{F}$$

into the category of **irredundant** words with functions. We have shown that the kernel of this functors is axiomatised by the equations of fig. 6.2:

Theorem 6.27. *The partially monoidal category* swF *modulo the equations of fig. 6.2 is isomorphic to the partially monoidal category* sw$\mathbb{F}$ *of* ***irredundant*** *words with functions.*

Proof. A quick inspection verifies that the left and right-hand side of all rewrites in fig. 6.2 are mapped to the same function between **irredundant** words in sw$\mathbb{F}$. This

[6] As discussed in sec. 6.5, we are currently not confident we have found all the critical peaks.
[7] https://goodlyrottenapple.github.io/string-diagrams-functions/confluence.html

shows the soundness of the equations. For completeness, we need to show that if two diagrams ϕ, ψ are identified by $[\![-]\!]$, then they can be proved equal using the equations. By lem. 6.23, we know that $[\![\phi]\!] = [\![\psi]\!]$ has a unique canonical form and by lems. 6.24, 6.26 we know that both ϕ and ψ rewrite to this canonical form. Hence ϕ and ψ are equal according to the equations.

□

Recall $[\![-]\!] : \mathsf{wF} \to \mathsf{w}\mathbb{F}$ from prop. 6.14.

Theorem 6.28. *Let ϕ, ψ be two* ***irredundant*** *diagrams in wF such that $\phi = \psi$ in the equational theory of wF plus the equations of fig. 6.2. Then $\phi = \psi$ in the equational theory of swF plus the equations of fig. 6.2.*

Proof. There are more equations in wF than in swF because the interchange law in swF is restricted by the partiality of parallel composition. Nevertheless, if $\phi = \psi$ in the equational theory of wF then $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{w}\mathbb{F}$ because the equations of fig. 6.2 are sound wrt to $[\![-]\!] : \mathsf{wF} \to \mathsf{w}\mathbb{F}$. But since ϕ and ψ are **irredundant**, $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{w}\mathbb{F}$ implies $[\![\phi]\!] = [\![\psi]\!]$ in $\mathsf{sw}\mathbb{F}$. Now the result follows from the completeness part of thm. 6.27.

□

6.5 Software Tools

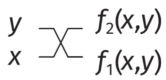
The proofs of termination and confluence presented in sec. 6.4 were given in reduced detail as the specifics are rather technical. In order to alleviate the burden, we developed software tools which helped building these proofs. These tools are presented briefly below.

6.5.1 Termination Proof

The termination proof in the previous section (lem. 6.24) uses a polynomial interpretation argument adapted from Lafont's original proof [49]. Whilst trying to modify the original proof (by adding the lollipop generator and the associated rules) it quickly became tedious to check if all 19 rules preserved the order. Moreover, when playing around with different possible rules, one needs a way to quickly check whether the new rules still terminate or not. Since the polynomial interpretation involves only simple arithmetic, we decided to automate the proof checking by delegating this work to an SMT solver.

As a result, we developed a Python script, which encodes the generators and all the rules as first order logic formulas which are then given to the Z3 SMT solver to be verified.

All the generators are translated into abstract functions; for example,



becomes

```
f1 = Function('f1', IntSort(), IntSort(), IntSort())
f1_def = ForAll([x, y], f1(x, y) == x+y)
f2 = Function('f2', IntSort(), IntSort(), IntSort())
f2_def = ForAll([x, y], f2(x, y) == x)
```

The rules and the monotonicity condition

are then encoded as

```
And(
    f1(g(),x) >= x,
    f2(g(),x) >= g(),
    Or(f1(g(),x) > x, f2(g(),x) > g()))
```

which corresponds to $f_1(g(), x) \geq x \wedge f_2(g(), x) \geq g() \wedge (f_1(g(), x) > x \vee f_2(g(), x) > g())$, encoding the condition that at least one argument is strictly decreasing, and none are increasing. The implementation helped us to experiment with different functions f_1, f_2, g and was instrumental in finding the solution employed in the termination proof.

6.5.2 Confluence Proof

In order to check local confluence of our system, we decided to implement the rewrite system in Haskell. This allowed us to generate the confluence proofs automatically, by giving the tool the critical peak diagrams we wanted to check. In future work, we hope to also generate the critical peaks automatically and provide an argument that we have checked all the critical peaks for the given system, since, at the moment, we aren't sure whether we have indeed listed all the critical peaks.
Both the termination script and the implementation of the rewriting system can be found on github[8].

6.5.3 Related work

Having followed the style of proofs in [49], we have since been alerted to more modern and scalable approaches to proving termination and confluence of string diagram rewriting,

[8]https://github.com/goodlyrottenapple/string-diagrams-functions

described in [51, 52] and [53]. As we have found when implementing the confluence proof checker, using the naive approach of representing string diagrams in their traditional 1-D syntax does not scale particularly well. This is due to the large number of equivalent diagrams arising from the monoidal equations and makes rewriting diagrams in this form tricky and computationally expensive. The work referenced above uses a different data structure for string diagrams, namely *open hypergraphs*, which allow for much more simplified and efficient rewriting of diagrams. Several tools taking this or similar approaches have been developed to work with categories presented by 2-dimensional syntax in a graphical way. These include Globular [54], Quantomatic [55] and CARTOGRAPHER [56], the last of which is most closely aligned with our work, as it operates within the setting of symmetric monoidal categories. An interesting future direction might be to try to extend CARTOGRAPHER to work with nominal string diagrams.

Ut, quasi transactis sæpe omnibu rebu, profundant
Fluminis ingentes fluctus, vestemque cruentent.

Lucretius

7

Nominal string diagrams

As mentioned earlier, this chapter is an extended version of [45], which was presented at CALCO 2019. This paper is joint work with my supervisor Alexander Kurz.

7.1 Setting the Scene: String Diagrams and Nominal Sets

Whereas the last chapter focused on the presentation of nominal string diagrams via partially monoidal categories, we start this chapter with more background on the algebraic presentation of string diagrams. First, we review some of the terminology and basic definitions of ordinary string diagrams.

7.1.1 String Diagrams and PROPs

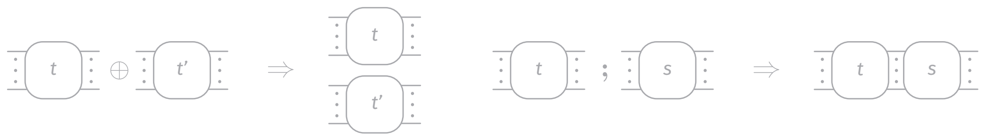
String diagrams are a 2-(or higher)-dimensional notation for monoidal categories [13]. Their algebraic theory can be formalised by PROPs as defined by MacLane [47].

A PROP (**pro**ducts and **p**ermutation category) is a symmetric strict monoidal category, with natural numbers as objects, where the monoidal tensor $\oplus$ is addition. Moreover, PROPs, along with strict symmetric monoidal functors, that are identities on objects, form the category PROP. A PROP contains all bijections between numbers as they can be be generated from the symmetry (twist) $\sigma : 1 \oplus 1 \to 1 \oplus 1$ and from the parallel composition $\oplus$ and sequential composition ; (which we write in diagrammatic order). We denote by $\sigma_{n,m}$ the canonical symmetry $n \oplus m \to m \oplus n$. Functors between PROPs preserve bijections.

PROPs can be presented in algebraic form by operations and equations as *symmetric monoidal theories* (SMTs) [57].

An SMT (Σ, E) has a set Σ of generators, where each generator $\gamma \in \Sigma$ is given an arity m and co-arity n, usually written as $\gamma : m \to n$ and a set E of equations, which are pairs of Σ-terms. Σ-terms can be obtained by composing generators in Σ with the unit $\mathbf{id} : 1 \to 1$ and symmetry $\sigma : 2 \to 2$, using either the parallel or sequential composition (see fig. 7.1). Equations E are pairs of Σ-terms with the same arity and co-arity.

$$\frac{}{\gamma : m \to n \in \Sigma} \qquad \frac{}{id : 1 \to 1} \qquad \frac{}{\sigma : 2 \to 2}$$



$$\frac{t : m \to n \qquad t' : o \to p}{t \oplus t' : m + o \to n + p} \qquad \frac{t : m \to n \qquad s : n \to o}{t \,;\, s : m \to o}$$

Figure 7.1: SMT Terms

Given an SMT $\langle \Sigma, E \rangle$, we can freely generate a PROP, by taking Σ-terms as arrows, modulo the equations **SMT**, which are:

- the equations stating that, together with **id**, the compositions ; and $\oplus$ form monoids
- the equations of fig. 7.2
- the equations E

$$\sigma_{1,1} \,;\, \sigma_{1,1} = id_2 \tag{SMT-sym}$$

$$(s \,;\, t) \oplus (u \,;\, v) = (s \oplus u) \,;\, (t \oplus v) \tag{SMT-ch}$$

$$\frac{s : m \to n \qquad t : o \to p}{(s \oplus t) \,;\, \sigma_{n,p} = \sigma_{m,o} \,;\, (t \oplus s)} \tag{SMT-nat}$$

Figure 7.2: Equations of symmetric monoidal categories

PROPs have a nice 2-dimensional notation, where sequential composition is horizontal composition of diagrams, and parallel/tensor composition is vertical stacking of diagrams (see fig. 7.1). We now present the SMTs of bijections **B**, injections **I**, surjections **S**,

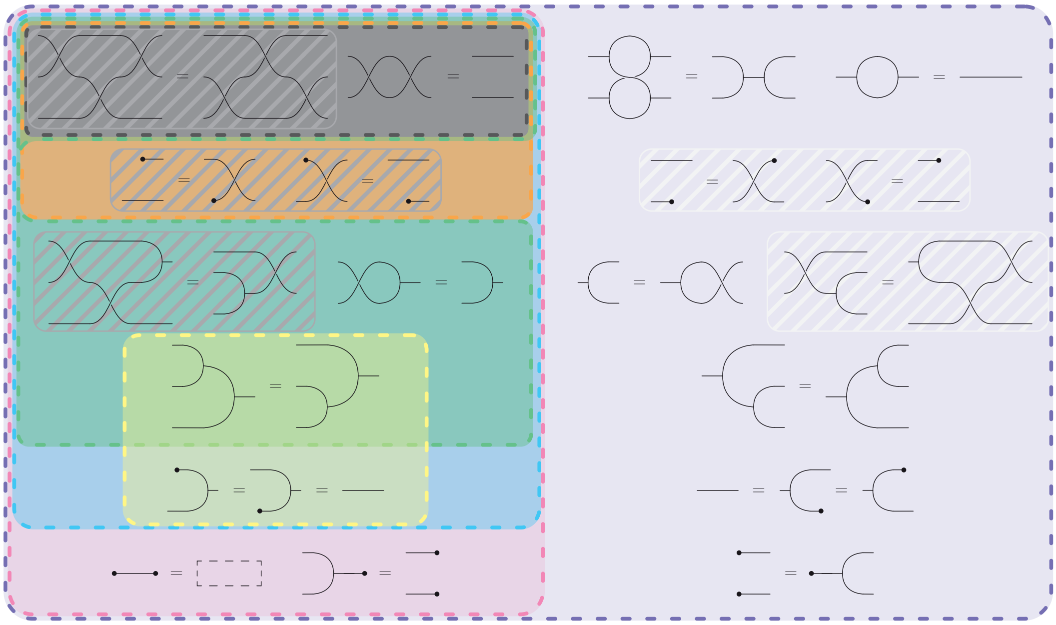


Figure 7.3: Symmetric monoidal theories (compiled from [49])

functions **F** , partial functions **P** , relations **R** and monotone maps **M** .[1]

The diagram in fig. 7.3 shows the generators and the equations that need to be added to the empty SMT, to get a presentation of the given theory.

To ease comparison with the corresponding nominal monoidal theories in fig. 7.4, we also added a striped background to the equations with wire-crossings, since they are already implied by the naturality of symmetries (SMT-nat). The right-hand equation for bijections **B** is (SMT-sym) and holds in all symmetric monoidal theories. We list it here to emphasise the difference with fig. 7.4.

7.1.2 Nominal Sets

Let $\mathcal{N}$ be a countably infinite set of 'names' or 'atoms'. Let $\mathfrak{S}$ be the group of finite[2] permutations $\mathcal{N} \to \mathcal{N}$. An element $x \in X$ of a group action $\mathfrak{S} \times X \to X$ is supported by $S \subseteq \mathcal{N}$ if $\pi \cdot x = x$ for all $\pi \in \mathfrak{S}$ such that π restricted to S is the identity. A group action $\mathfrak{S} \times X \to X$ where all elements of X have finite support is called a *nominal set*.

We write $\mathbf{supp}(x)$ for the minimal support of x and Nom for the category of nominal sets, which has as maps the *equivariant* functions, that is, those functions that respect the per-

[1]The theory of monotone maps **M** does not include equations involving the symmetry σ and is in fact presented by a so-called **PRO** rather than a PROP. However, in this paper we will only be dealing with theories presented by PROPs (the reason why this is the case is illustrated in the proof of prop. 7.29).

[2]A permutation is called finite if it is generated by finitely many transpositions.

mutation action. We present an example of a sub-category of $\underset{\sim}{\mathsf{Nom}}$; the category of simultaneous substitutions:

Example 7.1. [Category $\underset{\sim}{\mathsf{n}\mathbb{F}}$]

We denote by $\underset{\sim}{\mathsf{n}\mathbb{F}}$ the category of finite subsets of $\mathcal{N}$ with all functions. While $\underset{\sim}{\mathsf{n}\mathbb{F}}$ is a category, it also carries additional nominal structure. In particular, both the set of objects and the set of arrows are nominal sets with $\mathbf{supp}(A) = A$ and $\mathbf{supp}(f) = A \cup B$ for $f : A \to B$. The categories of injections $\mathsf{n}\mathbb{I}$, surjections $\mathsf{n}\mathbb{S}$, bijections $\mathsf{n}\mathbb{B}$, partial functions $\mathsf{n}\mathbb{P}$ and relations $\mathsf{n}\mathbb{R}$ are further examples along the same lines.

7.2 Internal monoidal categories

As stated in the introduction of this thesis, our exploration of string diagrams started out from the desire to create a calculus of simultaneous substitutions. Our aim was to give a presentation of the category $\underset{\sim}{\mathsf{n}\mathbb{F}}$ and we realised that string diagrams are an elegeant way to do just that.

Recall from rem. 6.6 that $\underset{\sim}{\mathbb{F}}$, which is presented by the $\underset{\sim}{\mathsf{SMT}}$ of functions **F** , is the skeleton category of $\underset{\sim}{\mathsf{n}\mathbb{F}}$. Whilst in $\underset{\sim}{\mathbb{F}}$, the monoidal tensor $\oplus$ is the coproduct

$$\oplus\colon \underset{\sim}{\mathbb{F}} \times \underset{\sim}{\mathbb{F}} \to \underset{\sim}{\mathbb{F}}$$

we see from the definition of $\underset{\sim}{\mathsf{n}\mathbb{F}}$ in ex. 7.1, that a monoidal product $\uplus$[3], corresponding to the parallel composition in $\underset{\sim}{\mathsf{n}\mathbb{F}}$, must be a partial operation

$$\uplus : \underset{\sim}{\mathsf{n}\mathbb{F}} \times \underset{\sim}{\mathsf{n}\mathbb{F}} \to \underset{\sim}{\mathsf{n}\mathbb{F}}.$$

This is the case, because we want to ensure there is no overlap between the domains and co-domains of the two functions we compose.

One way to formalise this is to develop a theory of partial monoidal categories, as we have done in the previous chapter. However, in this situation it seems more elegant to notice that $\uplus$ can be viewed as a total operation

$$\uplus : \underset{\sim}{\mathsf{n}\mathbb{F}} \circledast \underset{\sim}{\mathsf{n}\mathbb{F}} \to \underset{\sim}{\mathsf{n}\mathbb{F}}$$

if we take $\circledast$ as the "separated product" $A \circledast B = \{(a, b) \in A \times B \mid \mathbf{supp}(a) \cap \mathbf{supp}(b) = \varnothing\}$, which internalises the constraint that $f \uplus g$ is defined iff the domain and codomain of f

[3]We will use the $\uplus$ symbol for the monoidal tennsor of $\underset{\sim}{\mathsf{n}\mathbb{F}}$ to distingusih it from the monoidal tensor $\oplus$ of $\mathbb{F}$.

and g are disjoint. Whilst seemingly ad-hoc at first, we only have to look at the definition of Nom four our definition of $\circledast$:

> **Example 7.2.** Nom forms a symmetric monoidal (closed) category $(\mathrm{Nom}, 1, *)$ of nominal sets with the separated product $*$ (for details see [24]). 1 is the terminal object, i.e. a singleton with empty support. The separated product of two nominal sets is defined as $A * B = \{(a, b) \in A \times B \mid \mathbf{supp}(a) \cap \mathbf{supp}(b) = \varnothing\}$.

It is immediately obvious that our $\circledast$ is simply a “lifted” version of $*$ from Nom.

To make this lifting precise, we introduce the notion of an internal monoidal category. Given a symmetric monoidal category $(\mathcal{V}, I, \otimes)$ with finite limits, such as Nom in our example, we are interested in categories $\mathbb{C}$, internal in $\mathcal{V}$[4], that carry a monoidal structure not of type $\mathbb{C} \times \mathbb{C} \to \mathbb{C}$ but of type $\mathbb{C} \circledast \mathbb{C} \to \mathbb{C}$. Here we will make a distinction between the $\otimes$ monoidal product of the category $\mathcal{V}$

$$\otimes \colon \mathcal{V} \times \mathcal{V} \to \mathcal{V}$$

and the lifted product-of-internal-categories[5]

$$\circledast \colon \mathrm{Cat}(\mathcal{V}) \times \mathrm{Cat}(\mathcal{V}) \to \mathrm{Cat}(\mathcal{V}).$$

This lifted tensor will then allow us to account for the partiality of $\uplus$ discussed above, such that we have:

> **Example 7.3.** The category $(\mathrm{n}\mathbb{F}, \varnothing, \uplus)$ is an internal monoidal category (in Nom), with monoidal operation given by $A \uplus B = A \cup B$ if A, B are disjoint and $f \uplus f' = f \cup f'$ if A, A' and B, B' are disjoint where $f : A \to B$ and $f' : A' \to B'$.
>
> $(\mathrm{n}\mathbb{F}, \varnothing, \uplus)$ as defined in this example is not a monoidal category, since $\uplus$, is not an operation of type $\mathrm{n}\mathbb{F} \times \mathrm{n}\mathbb{F} \to \mathrm{n}\mathbb{F}$, but insted $\mathrm{n}\mathbb{F} \circledast \mathrm{n}\mathbb{F} \to \mathrm{n}\mathbb{F}$.

The purpose of this section is to give a proper definition of the notion of internal monoidal categories and to show that $(\mathrm{n}\mathbb{F}, \varnothing, \uplus)$ is an internal monoidal category in $(\mathrm{Nom}, 1, *)$.

From our example $\mathrm{n}\mathbb{F}$ above, we know that we want arrows (f, g) to be in $(\mathbb{C} \circledast \mathbb{C})_1$[6], if $\mathbf{dom}(f) \cap \mathbf{dom}(g) = \varnothing$ and $\mathbf{cod}(f) \cap \mathbf{cod}(g) = \varnothing$. One might be tempted to lift the tensor $\otimes$ from $\mathcal{V}$ in the obvious way: $(\mathbb{C} \circledast \mathbb{C})_1 = \mathbb{C}_1 \otimes \mathbb{C}_1$. However, since $\otimes$ need not preserve finite limits, we cannot expect that defining $(\mathbb{C} \circledast \mathbb{C})_0 = \mathbb{C}_0 \otimes \mathbb{C}_0$ and $(\mathbb{C} \circledast \mathbb{C})_1 = \mathbb{C}_1 \otimes \mathbb{C}_1$ results in $\mathbb{C} \circledast \mathbb{C}$ being an internal category. To show what goes wrong in a concrete instance, see the next example.

---

[4] For a definition of an internal category, see app. B

[5] In the type signature of $\circledast$, $\mathrm{Cat}(\mathcal{V})$ deonotes the category of small internal categories

[6] In this case, we have $\mathbb{C} = \mathrm{n}\mathbb{F}$.

Example 7.4. Following on from the previous example, given $(\underset{\sim}{\mathrm{Nom}}, 1, *)$, we define a binary operation $\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}}$ as $(\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}})_0 = \underset{\sim}{\mathrm{n}\mathbb{F}}_0 * \underset{\sim}{\mathrm{n}\mathbb{F}}_0$ and $(\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}})_1 = \underset{\sim}{\mathrm{n}\mathbb{F}}_1 * \underset{\sim}{\mathrm{n}\mathbb{F}}_1$. Then $\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}}$ cannot be equipped with the structure of an internal category. Indeed, assume for a contradiction that there was an appropriate pullback $(\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}})_2$ and arrow **comp** such that the two diagrams commute:

$$
\begin{tikzcd}
(\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}})_2 \arrow[rr, "\mathbf{comp}"] \arrow[d, "\pi_1"', "\pi_2"] & & \underset{\sim}{\mathrm{n}\mathbb{F}}_1 * \underset{\sim}{\mathrm{n}\mathbb{F}}_1 \arrow[d, "\mathbf{dom}"', "\mathbf{cod}"] \\
\underset{\sim}{\mathrm{n}\mathbb{F}}_1 * \underset{\sim}{\mathrm{n}\mathbb{F}}_1 \arrow[rr, "\mathbf{dom}", "\mathbf{cod}"'] & & \underset{\sim}{\mathrm{n}\mathbb{F}}_0 * \underset{\sim}{\mathrm{n}\mathbb{F}}_0
\end{tikzcd}
$$

Let $\delta_{xy} : \{x\} \to \{y\}$ be the unique function in $\underset{\sim}{\mathrm{n}\mathbb{F}}$ of type $\{x\} \to \{y\}$. Then $((\delta_{ac}, \delta_{bd}), (\delta_{cb}, \delta_{da}))$, which can be depicted as

$$
\begin{tikzcd}
\{a\} \arrow[r, "\delta_{ac}"] & \{c\} \arrow[r, "\delta_{cb}"] & \{b\} \\
\{b\} \arrow[r, "\delta_{bd}"'] & \{d\} \arrow[r, "\delta_{da}"'] & \{a\}
\end{tikzcd}
$$

is in the pullback $(\underset{\sim}{\mathrm{n}\mathbb{F}} \circledast \underset{\sim}{\mathrm{n}\mathbb{F}})_2$, but there is no **comp** such that the two squares above commute, since $\mathbf{comp}((\delta_{ac}, \delta_{bd}), (\delta_{cb}, \delta_{da}))$ would have to be $(\delta_{ab}, \delta_{ba})$. But since δ_{ab} and δ_{ba} do not have disjoint support (since $\mathbf{supp}(\delta_{ab}) = \mathbf{supp}(\delta_{ba}) = \{a, b\}$), this set cannot be in $\underset{\sim}{\mathrm{n}\mathbb{F}}_1 * \underset{\sim}{\mathrm{n}\mathbb{F}}_1$. □

The solution to the problem consists in assuming that the given symmetric monoidal category with finite limits $(\mathcal{V}, 1, \otimes)$ is semi-cartesian (aka affine), that is, the unit 1 is the terminal object. In such a category there are canonical arrows natural in A and B

$$j : A \otimes B \to A \times B$$

and we can use them to define arrows $j_1 : (\mathbb{C} \circledast \mathbb{C})_1 \to \mathbb{C}_1 \times \mathbb{C}_1$ that give us the right notion of tensor on arrows. We now turn this into a category theoretic definition, which is in fact an instance of the general and well-known construction of pulling back an internal category $\mathbb{C}$ along an arrow $j : X \to \mathbb{C}_0$. This construction yields an internal category $\mathbb{X}$ with $\mathbb{X}_0 = X$ and $\mathbb{X}_1$ the pullback of $\langle \mathbf{dom}_{\mathbb{C}}, \mathbf{cod}_{\mathbb{C}} \rangle$ along $j \times j$, or, equivalently, the limit in the following diagram

$$
\begin{tikzcd}
 & \mathbb{X}_1 \arrow[rr, "j_1"] \arrow[dl, "\mathbf{dom}_{\mathbb{X}}"'] \arrow[dr, dotted, "\mathbf{cod}_{\mathbb{X}}"] & & \mathbb{C}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{C}}"'] \arrow[dr, dotted, "\mathbf{cod}_{\mathbb{C}}"] & \\
 & & \mathbb{X}_0 \arrow[rr, dotted, "j"] & & \mathbb{C}_0 \\
\mathbb{X}_0 \arrow[rr, "j"'] & & \mathbb{C}_0 & &
\end{tikzcd}
$$

which we abbreviate to

$$
\begin{tikzcd}
\mathbb{X}_1 \arrow[r, "j_1"] \arrow[d, shift right, "\mathbf{dom}_{\mathbb{X}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{X}}"] & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}_{\mathbb{C}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{C}}"] \\
\mathbb{X}_0 \arrow[r, "j"'] & \mathbb{C}_0
\end{tikzcd}
\tag{7.1}
$$

Next we define $\boldsymbol{i} : \mathbb{X}_0 \to \mathbb{X}_1$ as the arrow into the limit $\mathbb{X}_1$ given by

$$
\begin{tikzcd}
\mathbb{X}_0 \arrow[dr, dotted, "i_{\mathbb{X}}"] \arrow[drr, bend left, "i_{\mathbb{C}} \circ j"] \arrow[ddr, bend right, shift left, "\mathrm{id}"] \arrow[ddr, bend right=50, "\mathrm{id}"'] & & \\
& \mathbb{X}_1 \arrow[r, "j_1"] \arrow[d, shift right, "\mathbf{dom}_{\mathbb{X}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{X}}"] & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}_{\mathbb{C}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{C}}"] \\
& \mathbb{X}_0 \arrow[r, "j"'] & \mathbb{C}_0
\end{tikzcd}
\tag{7.2}
$$

from which one reads off

$$
\mathbf{dom}_{\mathbb{X}} \circ i_{\mathbb{X}} = \mathrm{id}_{\mathbb{X}_0} = \mathbf{cod}_{\mathbb{X}} \circ i_{\mathbb{X}}
$$

Next, $\mathbb{X}_2$ is the pullback

$$
\begin{tikzcd}
& \mathbb{X}_2 \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{X}2}"] & \\
\mathbb{X}_1 \arrow[dr, "\mathbf{cod}_{\mathbb{X}}"'] & & \mathbb{X}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{X}}"] \\
& \mathbb{X}_0 &
\end{tikzcd}
$$

Recalling the definition of $\boldsymbol{j}_1$ from (7.1), there is also a corresponding $\boldsymbol{j}_2 : \mathbb{X}_2 \to \mathbb{C}_2$ due to the fact that the product of pullbacks is a pullback of products.

$$
\begin{tikzcd}
& \mathbb{X}_2 \arrow[rrr, "j_2"] \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{X}2}"] & & & \mathbb{C}_2 \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{C}2}"] & \\
\mathbb{X}_1 \arrow[rrr, dotted, bend left, "j_1"] \arrow[dr, "\mathbf{cod}_{\mathbb{X}}"'] & & \mathbb{X}_1 \arrow[rrr, dotted, bend right, "j_1"] \arrow[dl, "\mathbf{dom}_{\mathbb{X}}"] & \mathbb{C}_1 \arrow[dr, "\mathbf{cod}_{\mathbb{C}}"'] & & \mathbb{C}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{C}}"] \\
& \mathbb{X}_0 \arrow[rrr, "j"'] & & & \mathbb{C}_0 &
\end{tikzcd}
\tag{7.3}
$$

Recall the definition of the limit $\mathbb{X}_1$ from (7.1). Then $\mathbf{comp}_\mathbb{X} : \mathbb{X}_2 \to \mathbb{X}_1$ is the arrow into $\mathbb{X}_1$

$$\begin{tikzcd}
\mathbb{X}_2 \arrow[dr, dotted, "\mathbf{comp}_\mathbb{X}"] \arrow[drr, bend left, "\mathbf{comp}_\mathbb{C} \circ j_2"] \arrow[ddr, bend right, shift right, "\mathbf{dom}_\mathbb{X} \circ \pi_{\mathbb{X}1}"'] \arrow[ddr, bend right, shift left, "\mathbf{cod}_\mathbb{X} \circ \pi_{\mathbb{X}2}"] & & \\
& \mathbb{X}_1 \arrow[r, "j_1"] \arrow[d, shift right, "\mathbf{dom}_\mathbb{X}"'] \arrow[d, shift left, "\mathbf{cod}_\mathbb{X}"] & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}_\mathbb{C}"'] \arrow[d, shift left, "\mathbf{cod}_\mathbb{C}"] \\
& \mathbb{X}_0 \arrow[r, "j"'] & \mathbb{C}_0
\end{tikzcd} \tag{7.4}$$

from which one reads off

$$\mathbf{dom}_\mathbb{X} \circ \mathbf{comp}_\mathbb{X} = \mathbf{dom}_\mathbb{X} \circ \pi_{\mathbb{X}1} \qquad \mathbf{cod}_\mathbb{X} \circ \mathbf{comp}_\mathbb{X} = \mathbf{cod}_\mathbb{X} \circ \pi_{\mathbb{X}2} \qquad j_1 \circ \mathbf{comp}_\mathbb{X} = \mathbf{comp}_\mathbb{C} \circ j_2$$

and the remaining equations $\mathbf{comp}_\mathbb{X} \circ \langle i_\mathbb{X} \circ \mathbf{dom}_\mathbb{X}, \mathrm{id}_{\mathbb{X}_1} \rangle = \mathrm{id}_{\mathbb{X}_1} = \mathbf{comp}_\mathbb{X} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_\mathbb{X} \circ \mathbf{cod}_\mathbb{X} \rangle$ are also not difficult to prove.

Finally, by analogy with the definition of j_2 in (7.3), j_3 is defined as the unique arrow into the pullback $\mathbb{C}_3$, where $\mathbb{X}_3$ is defined in the expected way:

$$\begin{tikzcd}
& \mathbb{X}_3 \arrow[rrr, "j_3"] \arrow[dl, "\mathbf{left}_\mathbb{X}"'] \arrow[dr, "\mathbf{right}_\mathbb{X}"] & & & \mathbb{C}_3 \arrow[dl, "\mathbf{left}_\mathbb{C}"'] \arrow[dr, "\mathbf{right}_\mathbb{C}"] & \\
\mathbb{X}_2 \arrow[rrr, dotted, bend left, "j_2"] \arrow[dr, "\pi_{\mathbb{X}2}"'] & & \mathbb{X}_2 \arrow[rrr, dotted, bend right, "j_2"] \arrow[dl, "\pi_{\mathbb{X}1}"] & \mathbb{C}_2 \arrow[dr, "\pi_{\mathbb{C}2}"'] & & \mathbb{C}_2 \arrow[dl, "\pi_{\mathbb{C}1}"] \\
& \mathbb{X}_1 \arrow[rrr, "j_1"'] & & & \mathbb{C}_1 &
\end{tikzcd} \tag{7.5}$$

The equation $\mathbf{comp} \circ \mathbf{compl} = \mathbf{comp} \circ \mathbf{compr}$ will be shown in prop. 7.7.

This ends the definition of $\mathbb{X}$ as an internal category.

> *Note.* The notion of an internal category pulled back along some j has been formalised in the Lean theorem prover, with the proofs available on github[7].

To prove the next propositions, we will need the following lemma, which can be skipped for now. It is a consequence of the general fact that the isomorphism $[\mathcal{I}, \mathcal{C}](K_A, D) \cong \mathcal{C}(A, \lim D)$, defining limits, is natural in A and D.

[7] https://github.com/goodlyrottenapple/lean-internal-cats

Lemma 7.5. *If in the diagram*

$$
\begin{tikzcd}
 & T \arrow[rrr, "k"] \arrow[d, dotted, "h"] \arrow[dl, bend right, "f_1"'] \arrow[dr, bend left, "f_2"] & & & U \arrow[d, dotted, "h'"] \arrow[dl, bend right, "f'_1"'] \arrow[dr, bend left, "f'_2"] & \\
 & \mathbb{X}_2 \arrow[rrr, "j_2"] \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{X}2}"] & & & \mathbb{C}_2 \arrow[dl, "\pi_{\mathbb{C}1}"'] \arrow[dr, "\pi_{\mathbb{C}2}"] & \\
\mathbb{X}_1 \arrow[dr, "\mathbf{cod}_{\mathbb{X}}"'] \arrow[rrr, dotted, bend left, "j_1"] & & \mathbb{X}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{X}}"] \arrow[rrr, dotted, bend right, "j_1"] & \mathbb{C}_1 \arrow[dr, "\mathbf{cod}_{\mathbb{C}}"'] & & \mathbb{C}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{C}}"] \\
 & \mathbb{X}_0 \arrow[rrr, "j"'] & & & \mathbb{C}_0 &
\end{tikzcd}
$$

f_i and f'_i are cones commuting with j_1 and k, that is, if

$$\mathbf{cod}_{\mathbb{X}} \circ f_1 = \mathbf{dom}_{\mathbb{X}} \circ f_2 \tag{7.6}$$

$$\mathbf{cod}_{\mathbb{C}} \circ f'_1 = \mathbf{dom}_{\mathbb{C}} \circ f'_2 \tag{7.7}$$

$$j_1 \circ f_i = f'_i \circ k \tag{7.8}$$

and h, h' are the respective unique arrows into the pullbacks, then also

$$h' \circ k = j_2 \circ h$$

holds.

Proof. It suffices to calculate

$$\pi_{\mathbb{C}i} \circ h' \circ k = f'_i \circ k = j_1 \circ f_i = j_1 \circ \pi_{\mathbb{X}i} \circ h = \pi_{\mathbb{C}i} \circ j_2 \circ h$$

This implies $h' \circ k = j_2 \circ h$, due to the uniqueness of arrows into a limit. In full detail, we have:

$$\pi_{\mathbb{C}i} \circ h' \circ k = f'_i \circ k$$

which follow from the fact that h' is the unique arrow into a pullback, therefore $\pi_{\mathbb{C}i} \circ h' = f'_i$,

$$f'_i \circ k = j_1 \circ f_i$$

follow from (7.8),

$$j_1 \circ f_i = j_1 \circ \pi_{\mathbb{X}i} \circ h$$

follow from the fact that h is the unique arrow into a pullback, therefore $\pi_{\mathbb{X}i} \circ h = f_i$, and finally

$$j_1 \circ \pi_{\mathbb{X}i} \circ h = \pi_{\mathbb{C}i} \circ j_2 \circ h$$

which follow due to the equations $j_1 \circ \pi_{\mathbb{X}i} = \pi_{\mathbb{C}i} \circ j_2$, which can be read off from the definition of j_2, given in (7.3).

$\square$

Proposition 7.6. $\mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = \mathrm{id}_{\mathbb{X}_1} = \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle$.

Proof. We show the first equality $\mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = \mathrm{id}_{\mathbb{X}_1}$. According to the definition of the limit $\mathbb{X}_1$

$$\begin{tikzcd}
\mathbb{X}_1 \arrow[dr, dotted, "\mathrm{id}_{\mathbb{X}}"] \arrow[drr, bend left, "j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle"] \arrow[ddr, bend right, "\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle"] \arrow[ddr, bend right=50, "\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle"'] & & \\
& \mathbb{X}_1 \arrow[r, "j_1"] \arrow[d, shift right, "\mathbf{dom}_{\mathbb{X}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{X}}"] & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}_{\mathbb{C}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{C}}"] \\
& \mathbb{X}_0 \arrow[r, "j"'] & \mathbb{C}_0
\end{tikzcd}$$

it suffices to show

$$\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = \mathbf{dom}_{\mathbb{X}}$$

$$\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = \mathbf{cod}_{\mathbb{X}}$$

$$j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = j_1$$

The first two equalities follow from (7.4), namely $\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1}$ and $\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2}$.

$$\begin{aligned}
\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle &= \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle \\
&= \mathbf{dom}_{\mathbb{X}} \circ i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}} \\
&= \mathrm{id}_{\mathbb{X}_0} \circ \mathbf{dom}_{\mathbb{X}} \\
&= \mathbf{dom}_{\mathbb{X}}
\end{aligned}$$

$$\begin{aligned}
\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle &= \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle \\
&= \mathbf{cod}_{\mathbb{X}} \circ \mathrm{id}_{\mathbb{X}_1} \\
&= \mathbf{cod}_{\mathbb{X}}
\end{aligned}$$

$$
\begin{aligned}
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle &= \mathbf{comp}_{\mathbb{C}} \circ j_2 \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle \\
&= \mathbf{comp}_{\mathbb{C}} \circ \langle i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}}, \mathrm{id}_{\mathbb{C}_1} \rangle \circ j_1 \\
&= j_1
\end{aligned}
$$

In the last equation, we have $j_1 \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{comp}_{\mathbb{C}} \circ j_2$ by definition of **comp**, see (7.4). To prove $j_2 \circ \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle = \langle i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}}, \mathrm{id}_{\mathbb{C}_1} \rangle \circ j_1$, we instantiate lem. 7.5 with:

$$
\begin{aligned}
k &= j_1 & & \\
f_1 &= i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}} & f_1' &= i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}} \\
f_2 &= \mathrm{id}_{\mathbb{X}_1} & f_2' &= \mathrm{id}_{\mathbb{C}_1} \\
h &= \langle i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}}, \mathrm{id}_{\mathbb{X}_1} \rangle & h' &= \langle i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}}, \mathrm{id}_{\mathbb{C}_1} \rangle
\end{aligned}
$$

Instantiating the equations (7.6)-(7.8) with the data above, we need to show:

$$
\begin{aligned}
\mathbf{cod}_{\mathbb{X}} \circ i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}} &= \mathbf{dom}_{\mathbb{X}} \circ \mathrm{id}_{\mathbb{X}_1} \\
\mathbf{cod}_{\mathbb{C}} \circ i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}} &= \mathbf{dom}_{\mathbb{C}} \circ \mathrm{id}_{\mathbb{C}_1} \\
j_1 \circ i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}} &= i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}} \circ j_1 \\
j_1 \circ \mathrm{id}_{\mathbb{X}_1} &= \mathrm{id}_{\mathbb{C}_1} \circ j_1
\end{aligned}
$$

The first two equations follow from $\mathbf{cod}_{\mathbb{X}} \circ i_{\mathbb{X}} = \mathrm{id}_{\mathbb{X}_0}$ and $\mathbf{cod}_{\mathbb{C}} \circ i_{\mathbb{C}} = \mathrm{id}_{\mathbb{C}_0}$, see (7.2). The third follows from (7.2) and (7.1):

$$
\begin{aligned}
j_1 \circ i_{\mathbb{X}} \circ \mathbf{dom}_{\mathbb{X}} &= i_{\mathbb{C}} \circ j \circ \mathbf{dom}_{\mathbb{X}} \\
&= i_{\mathbb{C}} \circ \mathbf{dom}_{\mathbb{C}} \circ j_1
\end{aligned}
$$

and the last equality is trivial.

Now, for the other equality, $\mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle = \mathrm{id}_{\mathbb{X}_1}$. Again we need to show

$$
\begin{aligned}
\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= \mathbf{dom}_{\mathbb{X}} \\
\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= \mathbf{cod}_{\mathbb{X}} \\
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= j_1
\end{aligned}
$$

The first two equalities follow from (7.4), namely $\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1}$ and

$\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2}$.

$$\begin{aligned}
\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle \\
&= \mathbf{dom}_{\mathbb{X}} \circ \mathrm{id}_{\mathbb{X}_1} \\
&= \mathbf{dom}_{\mathbb{X}}
\end{aligned}$$

$$\begin{aligned}
\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle \\
&= \mathbf{cod}_{\mathbb{X}} \circ i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \\
&= \mathrm{id}_{\mathbb{X}_0} \circ \mathbf{cod}_{\mathbb{X}} \\
&= \mathbf{cod}_{\mathbb{X}}
\end{aligned}$$

$$\begin{aligned}
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle &= \mathbf{comp}_{\mathbb{C}} \circ j_2 \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle \\
&= \mathbf{comp}_{\mathbb{C}} \circ \langle \mathrm{id}_{\mathbb{C}_1}, i_{\mathbb{C}} \circ \mathbf{cod}_{\mathbb{C}} \rangle \circ j_1 \\
&= j_1
\end{aligned}$$

To prove $j_2 \circ \langle \mathrm{id}_{\mathbb{X}_1}, i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \rangle = \langle \mathrm{id}_{\mathbb{C}_1}, i_{\mathbb{C}} \circ \mathbf{cod}_{\mathbb{C}} \rangle \circ j_1$, we use lem. 7.5, checking:

$$\begin{aligned}
\mathbf{cod}_{\mathbb{X}} \circ \mathrm{id}_{\mathbb{X}_1} &= \mathbf{dom}_{\mathbb{X}} \circ i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} \\
\mathbf{cod}_{\mathbb{C}} \circ \mathrm{id}_{\mathbb{C}_1} &= \mathbf{dom}_{\mathbb{C}} \circ i_{\mathbb{C}} \circ \mathbf{cod}_{\mathbb{C}} \\
j_1 \circ i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} &= i_{\mathbb{C}} \circ \mathbf{cod}_{\mathbb{C}} \circ j_1 \\
j_1 \circ \mathrm{id}_{\mathbb{X}_1} &= \mathrm{id}_{\mathbb{C}_1} \circ j_1
\end{aligned}$$

where the first two equations follow from $\mathbf{dom}_{\mathbb{X}} \circ i_{\mathbb{X}} = \mathrm{id}_{\mathbb{X}_0}$ and $\mathbf{dom}_{\mathbb{C}} \circ i_{\mathbb{C}} = \mathrm{id}_{\mathbb{C}_0}$, see (7.2) and the third follows from (7.2) and (7.1):

$$\begin{aligned}
j_1 \circ i_{\mathbb{X}} \circ \mathbf{cod}_{\mathbb{X}} &= i_{\mathbb{C}} \circ j \circ \mathbf{cod}_{\mathbb{X}} \\
&= i_{\mathbb{C}} \circ \mathbf{cod}_{\mathbb{C}} \circ j_1
\end{aligned}$$

and the last equality is, again, trivial.

□

Proposition 7.7. $\mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} = \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}$

Proof. To show that composition is associative, we need to recall the definition of

compl and **compr** from def. B.1, which leads us to consider

$$
\begin{tikzcd}
\mathbb{X}_3 \arrow[dr, dotted, shift left, "\mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}}"] \arrow[dr, dotted, shift right, "\mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}"'] \arrow[drr, bend left, "j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}}"] \arrow[drr, bend left=15, "j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}"'] \arrow[ddr, bend right=10, "\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}}"] \arrow[ddr, bend right=20, "\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}"'] \arrow[ddr, bend right=30, "\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}}"] \arrow[ddr, bend right=40, "\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}"'] & & \\
& \mathbb{X}_1 \arrow[r, "j_1"] \arrow[d, shift right, "\mathbf{dom}_{\mathbb{X}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{X}}"] & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}_{\mathbb{C}}"'] \arrow[d, shift left, "\mathbf{cod}_{\mathbb{C}}"] \\
& \mathbb{X}_0 \arrow[r, "j"'] & \mathbb{C}_0
\end{tikzcd}
$$

To show $\mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} = \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}$ it suffices to show

$$
\begin{aligned}
\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}} \\
\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}} \\
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}
\end{aligned}
$$

For the first, we calculate

$$
\begin{aligned}
\mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \circ \mathbf{compl}_{\mathbb{X}} && \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \\
&= \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{left}_{\mathbb{X}} && \text{Def of } \mathbf{compl} \\
&= \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \circ \mathbf{left}_{\mathbb{X}} && \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \\
&= \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1} \circ \mathbf{compr}_{\mathbb{X}} && \text{Def of } \mathbf{compr} \\
&= \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}} && \mathbf{dom}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{dom}_{\mathbb{X}} \circ \pi_{\mathbb{X}1}
\end{aligned}
$$

and the second is similar:

$$
\begin{aligned}
\mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \circ \mathbf{compl}_{\mathbb{X}} && \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \\
&= \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \circ \mathbf{right}_{\mathbb{X}} && \text{Def of } \mathbf{compl} \\
&= \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{right}_{\mathbb{X}} && \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \\
&= \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2} \circ \mathbf{compr}_{\mathbb{X}} && \text{Def of } \mathbf{compr} \\
&= \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}} && \mathbf{cod}_{\mathbb{X}} \circ \mathbf{comp}_{\mathbb{X}} = \mathbf{cod}_{\mathbb{X}} \circ \pi_{\mathbb{X}2}
\end{aligned}
$$

The third, proceeding as in the proof of prop. 7.6, follows once we establish that the following commute:

$$
j_2 \circ \mathbf{compl}_{\mathbb{X}} = \mathbf{compl}_{\mathbb{C}} \circ j_3 \tag{7.9}
$$

$$
j_2 \circ \mathbf{compr}_{\mathbb{X}} = \mathbf{compr}_{\mathbb{C}} \circ j_3 \tag{7.10}
$$

But these two equations are again instances of lem. 7.5.

$$
\begin{tikzcd}
& \mathbb{X}_3 \arrow[rrrr, "j_3"] \arrow[d, dotted, "\mathbf{compl}_{\mathbb{X}}"] \arrow[ddl, bend right, "\mathbf{comp}_{\mathbb{X}} \circ \mathbf{left}_{\mathbb{X}}"'] \arrow[ddr, bend left, "\pi_{\mathbb{X}2} \circ \mathbf{right}_{\mathbb{X}}"] & & & & \mathbb{C}_3 \arrow[d, dotted, "\mathbf{compl}_{\mathbb{C}}"] \arrow[ddl, bend right, "\mathbf{comp}_{\mathbb{C}} \circ \mathbf{left}_{\mathbb{C}}"'] \arrow[ddr, bend left, "\pi_{\mathbb{C}2} \circ \mathbf{right}_{\mathbb{C}}"] & \\
& \mathbb{X}_2 \arrow[rrrr, "j_2"] \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{X}2}"] & & & & \mathbb{C}_2 \arrow[dl, "\pi_{\mathbb{X}1}"'] \arrow[dr, "\pi_{\mathbb{C}2}"] & \\
\mathbb{X}_1 \arrow[rrrr, dotted, bend left, "j_1"] \arrow[dr, "\mathbf{cod}_{\mathbb{X}}"'] & & \mathbb{X}_1 \arrow[rrrr, dotted, bend right, "j_1"'] \arrow[dl, "\mathbf{dom}_{\mathbb{X}}"] & & \mathbb{C}_1 \arrow[dr, "\mathbf{cod}_{\mathbb{C}}"'] & & \mathbb{C}_1 \arrow[dl, "\mathbf{dom}_{\mathbb{C}}"] \\
& \mathbb{X}_0 \arrow[rrrr, "j"'] & & & & \mathbb{C}_0 &
\end{tikzcd}
$$

Instantiating the diagram for the first equation, we only have to check that $j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{left}_{\mathbb{X}} = \mathbf{comp}_{\mathbb{C}} \circ \mathbf{left}_{\mathbb{C}} \circ j_3$ and $j_1 \circ \pi_{\mathbb{X}2} \circ \mathbf{right}_{\mathbb{X}} = \pi_{\mathbb{C}2} \circ \mathbf{right}_{\mathbb{C}} \circ j_3$, as the other equations follow from the respective definitions of $\mathbf{compl}_{\mathbb{X}}$ and $\mathbf{compl}_{\mathbb{C}}$.

$$
\begin{aligned}
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{left}_{\mathbb{X}} &= \mathbf{comp}_{\mathbb{C}} \circ j_2 \circ \mathbf{left}_{\mathbb{X}} && \text{By (7.4)} \\
&= \mathbf{comp}_{\mathbb{C}} \circ \mathbf{left}_{\mathbb{C}} \circ j_3 && \text{By (7.5)}
\end{aligned}
$$

$$
\begin{aligned}
j_1 \circ \pi_{\mathbb{X}2} \circ \mathbf{right}_{\mathbb{X}} &= \pi_{\mathbb{C}2} \circ j_2 \circ \mathbf{right}_{\mathbb{X}} && \text{By (7.3)} \\
&= \pi_{\mathbb{C}2} \circ \mathbf{right}_{\mathbb{C}} \circ j_3 && \text{By (7.5)}
\end{aligned}
$$

The proof of (7.10) follows in the same fashion.

Having proven equations (7.9) and (7.10), we show the final equality:

$$
\begin{aligned}
j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compl}_{\mathbb{X}} &= \mathbf{comp}_{\mathbb{C}} \circ j_2 \circ \mathbf{compl}_{\mathbb{X}} \\
&= \mathbf{comp}_{\mathbb{C}} \circ \mathbf{compl}_{\mathbb{C}} \circ j_3 \\
&= \mathbf{comp}_{\mathbb{C}} \circ \mathbf{compr}_{\mathbb{C}} \circ j_3 \\
&= \mathbf{comp}_{\mathbb{C}} \circ j_2 \circ \mathbf{compr}_{\mathbb{X}} \\
&= j_1 \circ \mathbf{comp}_{\mathbb{X}} \circ \mathbf{compr}_{\mathbb{X}}
\end{aligned}
$$

We have shown that composition is associative.

□

We have seen that the pullback of an internal category $\mathbb{C}$ along an arrow j with codomain $\mathbb{C}_0$ is an internal category:

Proposition 7.8. *Given an internal category $\mathbb{C}$ and an arrow $j : X \to \mathbb{C}_0$ there is an internal category $\mathbb{X}$ and an internal functor $\mathbb{j} : \mathbb{X} \to \mathbb{C}$ such that $\mathbb{X}_0 = X$ and $\mathbb{j}_0 = j$.*

Moreover, this internal category $\mathbb{X}$, or rather $\mathbb{j} : \mathbb{X} \to \mathbb{C}$, has a universal property known as a cartesian lifting. To make this precise, we recall the notion of a fibred category, or fibration.

Definition 7.9. [Fibration [58, 59]]

If $P : \mathcal{W} \to \mathcal{V}$ is a functor, then $\mathbb{j} : \mathbb{X} \to \mathbb{Y}$ is a *cartesian lifting* of $j : X \to P\mathbb{Y}$ (where $j = P\mathbb{j}$) if for all $\mathbb{k} : \mathbb{W} \to \mathbb{Y}$ and all $h : P\mathbb{W} \to X$ with $P\mathbb{k} = j \circ h$ there is a unique $\mathbb{h} : \mathbb{W} \to \mathbb{X}$ such that $\mathbb{j} \circ \mathbb{h} = \mathbb{k}$ and $P\mathbb{h} = h$.

$$\begin{tikzcd}
\mathbb{W} \arrow[dr, dotted, "!\mathbb{h}"'] \arrow[drr, "\mathbb{k}"] & & \\
& \mathbb{X} \arrow[r, "\mathbb{j}"'] & \mathbb{Y}
\end{tikzcd}$$

$$\begin{tikzcd}
P\mathbb{W} \arrow[dr, "P\mathbb{h}=h"'] \arrow[drr, "P\mathbb{k}=j\circ h"] & & \\
& X \arrow[r, "j"'] & P\mathbb{Y}
\end{tikzcd}$$

Moreover, $P : \mathcal{W} \to \mathcal{V}$ is called a (Grothendieck) *fibration* if all $j : X \to P\mathbb{Y}$ have a cartesian lifting for all $\mathbb{Y}$ in $\mathcal{W}$. If $P : \mathcal{W} \to \mathcal{V}$ is a fibration, the subcategory of $\mathcal{W}$ that has as arrows the arrows $\mathbb{f}$ such that $P\mathbb{f} = \mathrm{id}_Y$ is called the *fibre* over Y.

The next lemma is a strengthening of prop. 7.8.

Lemma 7.10. *Let $\mathcal{V}$ be a category with finite limits. The forgetful functor* $\mathsf{Cat}(\mathcal{V}) \to \mathcal{V}$ *is a fibration.*

Proof. We have already shown how to lift $j : X \to \mathbb{C}_0$ to $\mathbb{j} : \mathbb{X} \to \mathbb{C}$. One can show that this is a cartesian lifting by drawing out the appropriate diagram. Namely, we have the forgetful functor $(-)_0 : \mathsf{Cat}(\mathcal{V}) \to \mathcal{V}$, which sends an internal category to its "object of objects", an internal category $\mathbb{X}$, $\mathbb{Y}$ and an internal functor $\mathbb{j}$ between them. Given another internal category $\mathbb{W}$ and an internal functor $\mathbb{k} : \mathbb{W} \to \mathbb{Y}$ and an arrow $h : \mathbb{W}_0 \to \mathbb{Y}_0$, s.t. $\mathbb{k}_0 = \mathbb{j}_0 \circ h$, we show there is a unique $\mathbb{h}$, s.t. $\mathbb{k} = \mathbb{j} \circ \mathbb{h}$. This essentially means we need to fill in the following diagram, such that all sub-diagrams commute:

$$
\begin{tikzcd}
\mathbb{W}_2 \arrow[d, bend right, "\pi_1"'] \arrow[d, "\mathsf{comp}"] \arrow[d, bend left, "\pi_2"] \arrow[rr, bend left, "\mathbb{k}_2"] & \mathbb{X}_2 \arrow[d, bend right, "\pi_1"'] \arrow[d, "\mathsf{comp}"] \arrow[d, bend left, "\pi_2"] \arrow[r, "\mathbb{j}_2"] & \mathbb{Y}_2 \arrow[d, bend right, "\pi_1"'] \arrow[d, "\mathsf{comp}"] \arrow[d, bend left, "\pi_2"] \\
\mathbb{W}_1 \arrow[d, shift right=2, "\mathsf{dom}_{\mathbb{W}}"'] \arrow[d, shift left=2, "\mathsf{cod}_{\mathbb{W}}"] \arrow[rr, bend right, "\mathbb{k}_1"] & \mathbb{X}_1 \arrow[d, shift right=2, "\mathsf{dom}_{\mathbb{X}}"'] \arrow[d, shift left=2, "\mathsf{cod}_{\mathbb{X}}"] \arrow[r, "\mathbb{j}_1"] & \mathbb{Y}_1 \arrow[d, shift right=2, "\mathsf{dom}_{\mathbb{Y}}"'] \arrow[d, shift left=2, "\mathsf{cod}_{\mathbb{Y}}"] \\
\mathbb{W}_0 \arrow[u, "i_{\mathbb{W}}"] \arrow[r, "h"'] \arrow[rr, bend right, "\mathbb{k}_0"'] & \mathbb{X}_0 \arrow[u, "i_{\mathbb{X}}"] \arrow[r, "\mathbb{j}_0"'] & \mathbb{Y}_0 \arrow[u, "i_{\mathbb{Y}}"]
\end{tikzcd}
$$

Since our category has all finite limits, we can define $\mathbb{h}_1$ as an arrow into the limit $\mathbb{X}_1$:

$$
\begin{tikzcd}
\mathbb{W}_1 \arrow[dr, dotted, "\mathbb{h}_1"] \arrow[drr, bend left, "\mathbb{k}_1"] \arrow[ddr, bend right, "\mathsf{cod}_{\mathbb{W}} \circ h"] \arrow[ddr, bend right=50, "\mathsf{dom}_{\mathbb{W}} \circ h"'] & & \\
& \mathbb{X}_1 \arrow[d, shift right=2, "\mathsf{dom}_{\mathbb{X}}"'] \arrow[d, shift left=2, "\mathsf{cod}_{\mathbb{X}}"] \arrow[r, "\mathbb{j}_1"] & \mathbb{Y}_1 \arrow[d, shift right=2, "\mathsf{dom}_{\mathbb{Y}}"'] \arrow[d, shift left=2, "\mathsf{cod}_{\mathbb{Y}}"] \\
& \mathbb{X}_0 \arrow[r, "\mathbb{j}_0"'] & \mathbb{Y}_0
\end{tikzcd}
$$

We obtain $\mathbb{h}_2$ in a similar fashion, thus getting a unique $\mathbb{h} = (\mathbb{h}_2, \mathbb{h}_1, h)$, for which we have $\mathbb{k} = \mathbb{j} \circ \mathbb{h}$.

□

Instantiating lem. 7.10 with $\mathbb{C} \times \mathbb{D} \in \mathsf{Cat}(\mathcal{V})$ for $\mathbb{Y}$ and $j : \mathbb{C}_0 \otimes \mathbb{D}_0 \to \mathbb{C}_0 \times \mathbb{D}_0$ for $j : \mathbb{X}_0 \to \mathbb{Y}_0$, gives us the desired result from the beginning of this section, namely that the tensor $\otimes$ in a symmetric monoidal category $(\mathcal{V}, 1, \otimes)$ can be lifted to a tensor $\boldsymbol{\otimes}$: $Cat(\mathcal{V}) \times Cat(\mathcal{V}) \to Cat(\mathcal{V})$:

Corollary 7.11. *The arrow $j : \mathbb{C}_0 \otimes \mathbb{D}_0 \to \mathbb{C}_0 \times \mathbb{D}_0$ lifts to a morphism of internal categories $\mathbb{j} : \mathbb{C} \boldsymbol{\otimes} \mathbb{D} \to \mathbb{C} \times \mathbb{D}$. Moreover, $\mathbb{j}$ is the cartesian lifting of j.*

To show that this construction is functorial we need to use that $\otimes$: $\mathcal{V} \times \mathcal{V} \to \mathcal{V}$ is functorial and that $j : \mathbb{C}_0 \otimes \mathbb{D}_0 \to \mathbb{C}_0 \times \mathbb{D}_0$ is natural in $\mathbb{C}$ and $\mathbb{D}$. In order to lift such natural transformations, which are arrows in the functor category $\mathcal{V}^{\mathsf{Cat}(\mathcal{V}) \times \mathsf{Cat}(\mathcal{V})}$, we use

Lemma 7.12. *If $P : \mathcal{W} \to \mathcal{V}$ is a fibration and $\mathcal{A}$ is a category, then $P^{\mathcal{A}} : \mathcal{W}^{\mathcal{A}} \to \mathcal{V}^{\mathcal{A}}$ is a fibration.*

Proof. $P^{\mathcal{A}}$ is defined via post-composition with P, that is, $P^{\mathcal{A}}(\mathbb{G}) = P \circ \mathbb{G} = P\mathbb{G}$ and $P^{\mathcal{A}}(\eta : \mathbb{G} \to \mathbb{H}) = P\eta$.

To show that $P^{\mathcal{A}}$ is a fibration, i.e.that all $j : G \to P\mathbb{H}$ have a *cartesian lifting* $\mathbb{j} : \mathbb{G} \to \mathbb{H}$, we lift j pointwise, using the fact that for all $j_A : GA \to P\mathbb{H}A$ we have

$$\mathbb{j}_A : \mathbb{G}A \to \mathbb{H}A$$

due to P being a fibration. It remains to check that $\mathbb{j}$ is a *cartesian lifting*, that is, given natural transformations $\mathbb{k} : \mathbb{F} \to \mathbb{H}$ and $h : P\mathbb{F} \to G$, such that $P\mathbb{k} = P\mathbb{j} \circ h$, there is a unique $\mathbb{h}$, s.t. the following diagrams commute:

$$\begin{tikzcd}
\mathbb{F} \arrow[dr, Rightarrow, dotted, "!\mathbb{h}"'] \arrow[drr, Rightarrow, bend left, "\mathbb{k}"] & & \\
& \mathbb{G} \arrow[r, Rightarrow, "\mathbb{j}"'] & \mathbb{H}
\end{tikzcd}$$

$$\begin{tikzcd}
P\mathbb{F} \arrow[dr, Rightarrow, "P\mathbb{h}=h"'] \arrow[drr, Rightarrow, bend left, "P\mathbb{k} = P\mathbb{j} \circ h"] & & \\
& G \arrow[r, Rightarrow, "j = P\mathbb{j}"'] & P\mathbb{H}
\end{tikzcd}$$

Since $\mathbb{k}$, $\mathbb{j}$ and h are natural transformations, i.e a family of morphisms, for any $A, B \in \mathcal{A}$ and $f : A \to B$, we have:

$$\begin{tikzcd}
\mathbb{F}A \arrow[r, "\mathbb{F}f"] \arrow[d, "\mathbb{k}_A"] & \mathbb{F}B \arrow[d, "\mathbb{k}_B"'] \\
\mathbb{H}A \arrow[r, "\mathbb{H}f"] & \mathbb{H}B \\
\mathbb{G}A \arrow[r, "\mathbb{G}f"'] \arrow[u, "\mathbb{j}_A"'] & \mathbb{G}B \arrow[u, "\mathbb{j}_B"]
\end{tikzcd}$$

$$\begin{tikzcd}
P\mathbb{F}A \arrow[r, "P\mathbb{F}f"] \arrow[d, "P\mathbb{k}_A"] \arrow[dd, bend right, "h_A"'] & P\mathbb{F}B \arrow[d, "P\mathbb{k}_B"'] \arrow[dd, bend left, "h_B"] \\
P\mathbb{H}A \arrow[r, "P\mathbb{H}f"] & P\mathbb{H}B \\
P\mathbb{G}A \arrow[r, "P\mathbb{G}f"'] \arrow[u, "P\mathbb{j}_A"'] & P\mathbb{G}B \arrow[u, "P\mathbb{j}_B"]
\end{tikzcd}$$

As P is a fibration, we obtain unique $\mathbb{h}_A$ and $\mathbb{h}_B$ for the left and right sub-diagrams above, s.t. $P\mathbb{h}_A = h_A$ and $P\mathbb{h}_B = h_B$, thus obtaining a unique natural transformation $\mathbb{h}$, for which $\mathbb{k} = \mathbb{h} \circ \mathbb{j}$.

□

Instantiating the lemma with $P = (-)_0 : \mathbf{Cat}(\mathcal{V}) \to \mathcal{V}$ and $\mathcal{A} = \mathbf{Cat}(\mathcal{V}) \times \mathbf{Cat}(\mathcal{V})$, we obtain as a corollary that lifting the tensor $\otimes\colon \mathcal{V} \times \mathcal{V} \to \mathcal{V}$ to $\boldsymbol{\otimes}\colon \mathit{Cat}(\mathcal{V}) \times \mathit{Cat}(\mathcal{V}) \to \mathit{Cat}(\mathcal{V})$ is functorial:

Theorem 7.13. *Let $(\mathcal{V}, 1, \otimes)$ be a (symmetric) monoidal category with finite limits in which the monoidal unit is the terminal object. Let $U : \mathrm{Cat}(\mathcal{V}) \to \mathcal{V}$ be the forgetful functor from categories internal in $\mathcal{V}$. Then the canonical arrow $j : \mathbb{C}_0 \otimes \mathbb{D}_0 \to \mathbb{C}_0 \times \mathbb{D}_0$ lifts to a natural transformation $\mathbb{j} : \mathbb{C} \boldsymbol{\otimes} \mathbb{D} \to \mathbb{C} \times \mathbb{D}$. Moreover, $(\mathrm{Cat}(\mathcal{V}), \mathbb{I}, \boldsymbol{\otimes})$ inherits from $(\mathcal{V}, 1, \otimes)$ the structure of a (symmetric) monoidal category with finite limits in which the monoidal unit is the terminal object.*

Proof. Let

$$\mathbb{G} : \mathrm{Cat}(\mathcal{V}) \times \mathrm{Cat}(\mathcal{V}) \to \mathrm{Cat}(\mathcal{V}) \qquad F : \mathcal{V} \times \mathcal{V} \to \mathcal{V} \qquad \mathbb{F} : \mathrm{Cat}(\mathcal{V}) \times \mathrm{Cat}(\mathcal{V}) \to \mathrm{Cat}(\mathcal{V})$$
$$\mathbb{G}(\mathbb{A}, \mathbb{B}) = \mathbb{A} \times \mathbb{B} \qquad F(X, Y) = X \otimes Y \qquad \mathbb{F}(\mathbb{A}, \mathbb{B}) = \mathbb{A} \boldsymbol{\otimes} \mathbb{B}$$

and $j : FU \to U\mathbb{G}$ the associated natural transformation. We also have, by definition, that $FU = U\mathbb{F}$, namely for all $\mathbb{A}, \mathbb{B}$ we have $\mathbb{A}_0 \otimes \mathbb{B}_0 = (\mathbb{A} \boldsymbol{\otimes} \mathbb{B})_0$. Therefore, j lifts to a natural transformation $\mathbb{j} : \mathbb{F} \to \mathbb{G}$ where $\mathbb{j}$ is a cartesian lifting of j by lem. 7.12. As a direct consquence, $\mathbb{F}$ must be a functor. $\square$

In this work we only need internal monoidal categories that are strict. In the same way as a strict monoidal category is a monoid in $(\mathrm{Cat}, 1, \times)$, an internal strict monoidal category is a monoid in $(\mathrm{Cat}(\mathcal{V}), \mathbb{I}, \boldsymbol{\otimes})$:

Definition 7.14. [Internal monoidal category]

Let $(\mathcal{V}, 1, \otimes)$ be a symmetric monoidal category with finite limits in which the monoidal unit is the terminal object and let $(\mathrm{Cat}(\mathcal{V}), \mathbb{I}, \boldsymbol{\otimes})$ be the induced symmetric monoidal category of internal categories in $\mathcal{V}$. A strict internal monoidal category $\mathbb{C}$ is a monoid $(\mathbb{C}, \varnothing, \odot)$ in $(\mathrm{Cat}(\mathcal{V}), \mathbb{I}, \boldsymbol{\otimes})$.

Remark 7.15. It may be useful to recap and catalogue the different tensors. The first one is the cartesian product $\times$ of categories, with the help of which we define a monoidal product $\otimes$ on a particular category $\mathcal{V}$ and then lift it to a monoidal product $\boldsymbol{\otimes}$ on the category of categories internal in $\mathcal{V}$. This then allows us to define on an internal category $\mathbb{C}$ a tensor $\odot$, which we also call an *internal tensor*:

$$\otimes\colon \mathcal{V} \times \mathcal{V} \to \mathcal{V}$$
$$\boldsymbol{\otimes}\colon \mathrm{Cat}(\mathcal{V}) \times \mathrm{Cat}(\mathcal{V}) \to \mathrm{Cat}(\mathcal{V})$$
$$\odot : \mathbb{C} \boldsymbol{\otimes} \mathbb{C} \to \mathbb{C}$$

Example 7.16. Picking up ex. 7.3 again, for the category $\underset{\sim}{n}\mathbb{F}$ of finite sets of names and functions, we choose the empty set for $\varnothing$ and for the internal tensor $\odot = \uplus$, the union of disjoint sets on objects and, on arrows, the union of functions with both disjoint domains and disjoint codomains.

Remark 7.17. In the classical case where $\mathcal{V} =$ Cat and both $\otimes$ and $\boldsymbol{\otimes}$ are the cartesian product, the interchange law for $\odot$ follows from $\odot$ being a functor. In the same way, in our more general situation, the interchange law for $\odot$ states that $\odot$ is an internal functor (B.2)

$$\begin{tikzcd}
(\mathbb{C} \boldsymbol{\otimes} \mathbb{C})_2 \arrow[r, "\mathbf{comp}_{\mathbb{C}\boldsymbol{\otimes}\mathbb{C}}"] \arrow[d, "\odot_2"'] & (\mathbb{C} \boldsymbol{\otimes} \mathbb{C})_1 \arrow[d, "\odot_1"] \\
\mathbb{C}_2 \arrow[r, "\mathbf{comp}_{\mathbb{C}}"'] & \mathbb{C}_1
\end{tikzcd}$$

Example 7.18. In the category $(\underset{\sim}{n}\mathbb{F}, \varnothing, \uplus)$ of finite sets of names and functions, see ex. 7.3, we have the interchange law

$$(\mathbb{f} \uplus \mathbb{g});(\mathbb{f}' \uplus \mathbb{g}') = (\mathbb{f};\mathbb{f}') \uplus (\mathbb{g};\mathbb{g}')$$

with the right-hand side being defined whenever the left-hand side is.

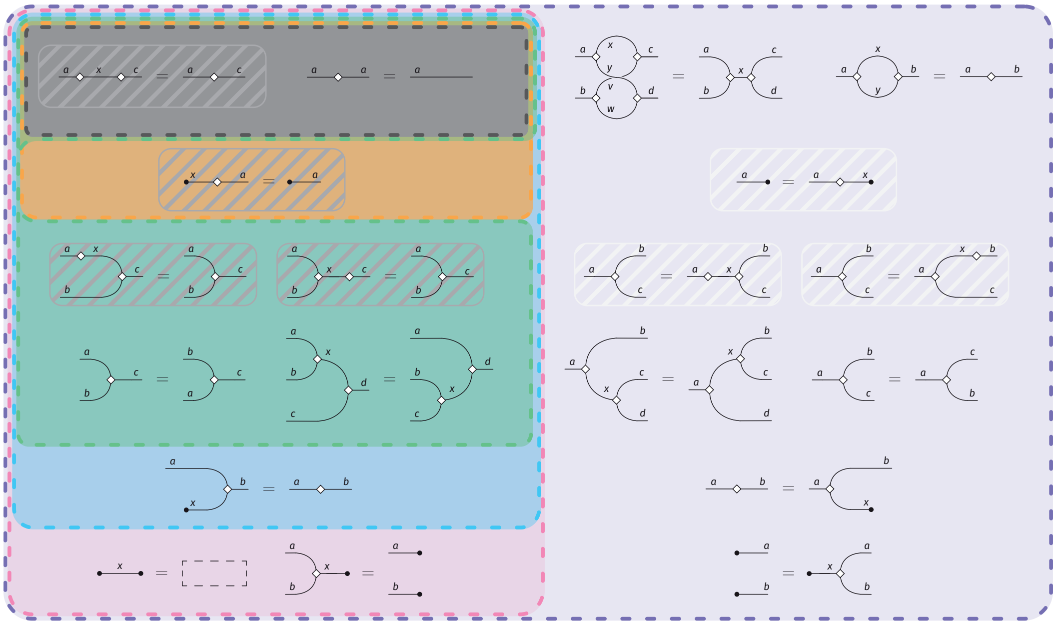
7.3 Examples

Before we give a formal definition of nominal PROPs and nominal monoidal theories (NMTs) in the next section, we present as examples those NMTs that correspond to the SMTs of fig. 7.3. The significant differences between fig. 7.3 and fig. 7.4 are that wires now carry labels and that there is a new generator $\overset{a}{—}\!\diamond\!\overset{b}{—}$ which allows us to change the label of a wire. Moreover, in the nominal setting rules for wire crossings are not needed.

The example below lists presentations of nominal monoidal theories for the nominal monoidal categories of finite sets and functions, injections, surjections, partial functions and relations, respectively.

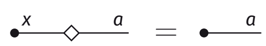
Example 7.19. The category of finite sets and

- bijections is presented by the empty signature and equations.
- injections is presented by $\Sigma_i = \{\eta_a : \varnothing \to \{a\} \mid a \in \mathcal{N}\}$ and $E_i = \varnothing$. The equations



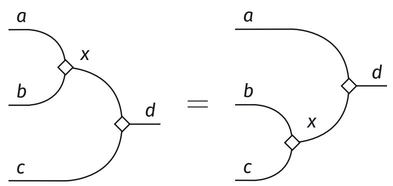
bijections nB , injections nI , surjections nS , functions nF , partial functions nP and relations nR

Figure 7.4: Nominal monoidal theories



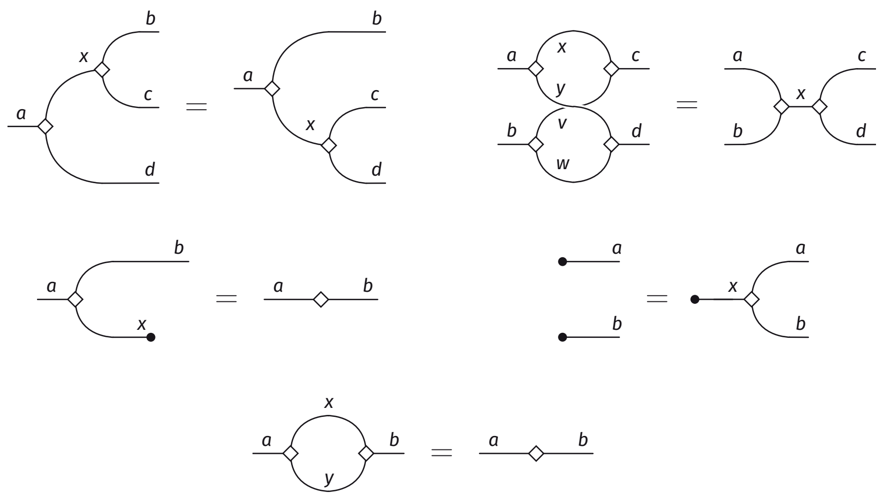
follow from those of fig. 7.8.

- surjections is presented by $\Sigma_s = \{\mu_{abc} : \{a, b\} \to \{c\} \mid a, b, c \in \mathcal{N}\}$ and equations E_s are $(\mu_{abx} \uplus id_c) \circ \mu_{cdx} = (\mu_{bcx} \uplus id_a) \circ \mu_{adx}$, presented graphically as

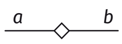


- functions has $\Sigma_f = \Sigma_i \cup \Sigma_s$ and equations E_f are $E_i \cup E_s$ plus $(id_a \uplus \eta_x) \circ \mu_{abx} = \delta_{ab}$
- partial functions has $\Sigma_{pf} = \Sigma_f \cup \{\hat{\eta}_a : \{a\} \to \varnothing \mid a \in \mathcal{N}\}$ and equations E_{pf} are E_f plus $\eta_x \circ \hat{\eta}_x = \varepsilon$ and $\mu_{abx} \circ \hat{\eta}_x = \hat{\eta}_a \uplus \hat{\eta}_b$, shown below

- relations has $\Sigma_r = \Sigma_{pf} \cup \{\hat{\mu}_{abc} : \{a\} \to \{b, c\} \mid a, b, c \in \mathcal{N}\}$, and equations E_r are E_{pf} plus the following



Theorem 7.20. *The calculi of fig. 7.4 are sound and complete, that is, the categories presented by these calculi are isomorphic to the categories of finite sets of names with the respective maps.*

We can prove this theorem in the same general fashion as the well-known proofs for SMTs (see e.g. Lafont [49]) and proceed by showing that each diagram $f : A \to B$ can be rewritten to one in normal form, with the normal form being a direct syntactic representation of the semantic function/relation represented by f. Such proofs for NMTs seem easier than the corresponding proofs for SMTs due to the absence of wire crossings. For example, in the case of bijections, it is immediate that, using the grey rules of fig. 7.4, every nominal diagram rewrites to a normal form which is just a parallel composition of diagrams of the form 

.

However, if we already have a soundness and completeness proof of an SMT for some semantic category, we can transfer this result over to an NMT, which presents the nominal version of this semantic category. For full details of this construction, see sec. 7.6.3.

7.4 Nominal monoidal theories and nominal PROPs

In this section, we introduce nominal PROPs as internal monoidal categories in nominal sets. We first spell out the details of what that means in elementary terms and then discuss the notion of diagrammatic α-equivalence.

7.4.1 Nominal monoidal theories

A *nominal monoidal theory* (Σ, E) is given by a set Σ of generators and a nominal set E of equations. The set of nominal generators $n\Sigma$ is generated by the set Σ of 'ordinary' generators $\gamma : n \to m$, each γ, giving rise to a set of nominal generators $[\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}] : A \to B$ where $\boldsymbol{a}, \boldsymbol{b}$ are unique lists of size n, m and whose underlying sets are A, B respectively. The nominal generators $n\Sigma$ are closed under permutations

$$\pi \cdot [\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}] : \pi \cdot A \to \pi \cdot B = [\pi(\boldsymbol{a})\rangle\gamma\langle\pi(\boldsymbol{b})]. \tag{π-def}$$

The set of nominal terms or nTrms is given by closing under the operations of fig. 7.5, which should be compared with fig. 7.1.

$$\frac{\gamma : m \to n \in \Sigma}{[\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}] : A \to B} \qquad \frac{}{id_a : \{a\} \to \{a\}} \qquad \frac{}{\delta_{ab} : \{a\} \to \{b\}}$$

$$\frac{t : A \to B \qquad t' : A' \to B'}{t \uplus t' : A \uplus A' \to B \uplus B'} \qquad \frac{t : A \to B \qquad s : B \to C}{t \,;\, s : A \to C} \qquad \frac{t : A \to B}{(a\ b)t : (a\ b) \cdot A \to (a\ b) \cdot B}$$

Figure 7.5: NMT Terms

Every NMT freely generates a monoidal category internal in nominal sets by quotienting the generated terms by equations in E, together with equations **NMT**:

- the equations that state that **id** and **;** obey the laws of a category
- the equations stating that $\mathbf{id}_\emptyset$ and $\uplus$ are a monoid
- the equations of an internal monoidal category of fig. 7.6
- the equations of permutation actions of fig. 7.7
- the equations on the interaction of generators with bijections δ of fig. 7.8[8]

$$t \uplus s = s \uplus t \tag{NMT-comm}$$
$$(s\ ;\ t) \uplus (u\ ;\ v) = (s \uplus u)\ ;\ (t \uplus v) \tag{NMT-ch}$$

Figure 7.6: NMT Equations of $\uplus$

$$(a\ b)id_x = id_{(a\ b)\cdot x} \qquad (a\ b)\delta_{xy} = \delta_{(a\ b)\cdot x\ (a\ b)\cdot y} \qquad (a\ b)\gamma = (a\ b)\cdot\gamma$$
$$(a\ b)(x \uplus y) = (a\ b)x \uplus (a\ b)y \qquad (a\ b)(x\ ;\ y) = (a\ b)x\ ;\ (a\ b)y$$

Figure 7.7: NMT Equations of the permutation actions

For terms to form a nominal set, we need equations between permutations to hold, along with the equations of fig. 7.7 that specify how permutations act on terms.

[8]The main difference with the equations in fig. 7.2 is that the interchange law for $\uplus$ is required to hold only if both sides are defined and that the two laws involving symmetries are replaced by the commutativity of $\uplus$.

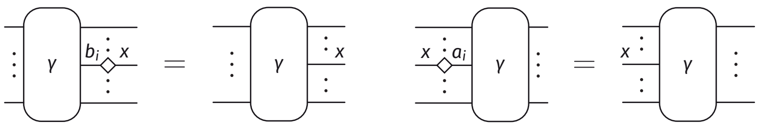
$$\delta_{aa} = \mathrm{id}_a \qquad\qquad \delta_{ab} \,;\, \delta_{bc} = \delta_{ac}$$

$$\frac{[a_1, \dots, a_i, \dots, a_m\rangle\gamma\langle\boldsymbol{b}] : \{a_i\} \uplus A \to B}{(\delta_{xa_i} \uplus \mathrm{id}_A) ; [a_1, \dots, a_i, \dots, a_m\rangle\gamma\langle\boldsymbol{b}] = [a_1, \dots, x, \dots, a_m\rangle\gamma\langle\boldsymbol{b}]} \quad \text{(NMT-left)}$$

$$\frac{[\boldsymbol{a}\rangle\gamma\langle b_1, \dots, b_i, \dots, b_n] : A \to B \uplus \{b_i\}}{[\boldsymbol{a}\rangle\gamma\langle b_1, \dots, b_i, \dots, b_n] ; (\mathrm{id}_B \uplus \delta_{b_i x}) = [\boldsymbol{a}\rangle\gamma\langle b_1, \dots, x, \dots, b_n]} \quad \text{(NMT-right)}$$

Figure 7.8: NMT Equations of δ

All the equations presented in the figures above are routine, with the exception of the last two, specifying the interaction of renamings δ with the generators $[\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}] \in \Sigma$, which we also depict in diagrammatic form:



Instances of these rules can be seen in fig. 7.4, where they are distinguished by a striped background.

7.4.2 Diagrammatic α-equivalence

The equations of fig. 7.7 and fig. 7.8 introduce a notion of *diagrammatic* $\boldsymbol{\alpha}$*-equivalence*, which allows us to rename 'internal' names and to contract renamings.

Definition 7.21. Two terms of a nominal monoidal theory are $\boldsymbol{\alpha}$-equivalent if their equality follows from the equations in fig. 7.7 and fig. 7.8.

Every permutation π of names gives rise to bijective functions $\pi_A : A \to \pi[A] = \{\pi(a) \mid a \in A\} = \pi \cdot A$. Any such π_A, as well as the inverse π_A^{-1}, are parallel compositions of δ_{ab} for suitable $a, b \in \mathcal{N}$. In fact, we have $\pi_A = \biguplus_{a \in A} \delta_{a\,\pi(a)}$ and $\pi_A^{-1} = \biguplus_{a \in A} \delta_{\pi(a)\,a}$. We may therefore use the π_A as abbreviations in terms.

Proposition 7.22. *Let $t : A \to B$ be a term of a nominal monoidal theory. The equations in fig. 7.7 and fig. 7.8 entail that $\pi \cdot t = \pi_A^{-1} \,;\, t \,;\, \pi_B$.*

$$\begin{tikzcd}
A \arrow[r, "t"] \arrow[d, "\pi_A"'] & B \arrow[d, "\pi_B"] \\
\pi[A] \arrow[r, "\pi \cdot t"'] & \pi[B]
\end{tikzcd}$$

Proof. By induction on t:

- $t = \mathrm{id}_a$ W.l.o.g. we can assume we have the following two cases. Either $\pi = (a\ x)$ or $\pi = (x\ y)$:

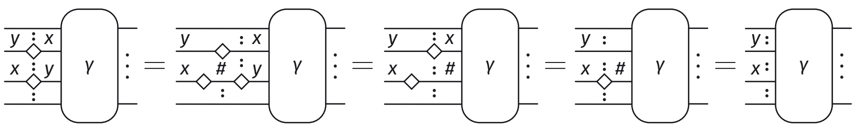
  - If $\pi = (a\ x)$, then
    $(a\ x)\mathrm{id}_a = \mathrm{id}_x = \delta_{xx} = \delta_{xa} ; \delta_{ax} = \delta_{xa} ; \mathrm{id}_a ; \delta_{ax} = \pi^{-1}_{\{a\}} ; \mathrm{id}_a ; \pi_{\{a\}}$
  - If $\pi = (x\ y)$, then
    $(x\ y)\mathrm{id}_a = \mathrm{id}_a = \mathrm{id}_a ; \mathrm{id}_a ; \mathrm{id}_a = \delta_{aa} ; \mathrm{id}_a ; \delta_{aa} = \pi^{-1}_{\{a\}} ; id_a ; \pi_{\{a\}}$

- $t = \delta_{ab}$ W.l.o.g. we can assume the following five cases for π:

  $$\pi = (a\ x) \text{ or } \pi = (b\ x) \text{ or } \pi = (a\ x)(b\ y) \text{ or } \pi = (a\ b) \text{ or } \pi = (x\ y)$$

  - If $\pi = (a\ x)$, then
    $(a\ x)\delta_{ab} = \delta_{xb} = \delta_{xa} ; \delta_{ab} = \delta_{xa} ; \delta_{ab} ; \delta_{bb} = \pi^{-1}_{\{a\}} ; \delta_{ab} ; \pi_{\{b\}}$
  - If $\pi = (b\ x)$, then
    $(b\ x)\delta_{ab} = \delta_{ax} = \delta_{ab} ; \delta_{bx} = \delta_{aa} ; \delta_{ab} ; \delta_{bx} = \pi^{-1}_{\{a\}} ; \delta_{ab} ; \pi_{\{b\}}$
  - If $\pi = (a\ x)(b\ y)$, then
    $(a\ x)(b\ y)\delta_{ab} = \delta_{xy} = \delta_{xb} ; \delta_{by} = \delta_{xa} ; \delta_{ab} ; \delta_{by} = \pi^{-1}_{\{a\}} ; \delta_{ab} ; \pi_{\{b\}}$
  - If $\pi = (a\ b)$, then
    $(a\ b)\delta_{ab} = \delta_{ba} = \mathrm{id}_b ; \delta_{ba} = \delta_{bb} ; \delta_{ba} = \delta_{ba} ; \delta_{ab} ; \delta_{ba} = \pi^{-1}_{\{a\}} ; \delta_{ab} ; \pi_{\{b\}}$
  - If $\pi = (x\ y)$, then
    $(x\ y)\delta_{ab} = \delta_{ab} = \delta_{aa} ; \delta_{ab} ; \delta_{bb} = \pi^{-1}_{\{a\}} ; \delta_{ab} ; \pi_{\{b\}}$

- $t = [\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}]$ The equality $\pi \cdot t = (\pi_A)^{-1} ; t ; \pi_B$ follows straightforwardly from repeated application of (NMT-left) and (NMT-right). The only problematic case arises in the case $\pi = (x\ y)$ and $t = [a_1, \dots, x, \dots, y, \dots a_n\rangle\gamma\langle\boldsymbol{b}]$ (there is a symmetric case with x, y on the right, which proceeds in exactly the same fashion):

  In the diagrams above, # is a fresh variable that does not appear in t.

- $t = s \uplus s'$ By IH we have $\pi \cdot s = \pi^{-1}_A ; s ; \pi_B$ and $\pi \cdot s' = \pi^{-1}_{A'} ; s ; \pi_{B'}$. We thus have:

$$
\begin{aligned}
\pi \cdot (s \uplus s') &= \pi \cdot s \uplus \pi \cdot s' \\
&= (\pi_A^{-1} ; s ; \pi_B) \uplus (\pi_{A'}^{-1} ; s' ; \pi_{B'}) \\
&= (\pi_A^{-1} \uplus \pi_{A'}^{-1}) ; ((s ; \pi_B) \uplus (; s' ; \pi_{B'})) \\
&= (\pi_A^{-1} \uplus \pi_{A'}^{-1}) ; (s \uplus s') ; (\pi_B \uplus \pi_{B'}) \\
&= \pi_{A \uplus A'}^{-1} ; (s \uplus s') ; \pi_{B \uplus B'}
\end{aligned}
$$

- $t = u ; v$ Again, we have $\pi \cdot u = \pi_A^{-1} ; u ; \pi_B$ and $\pi \cdot v = \pi_B^{-1} ; v ; \pi_C$ by IH. Then:

$$
\begin{aligned}
\pi \cdot (u ; v) &= \pi \cdot u ; \pi \cdot v \\
&= \pi_A^{-1} ; u ; \pi_B ; \pi_B^{-1} ; v ; \pi_C \\
&= \pi_A^{-1} ; u ; \mathrm{id}_B ; v ; \pi_C \\
&= \pi_A^{-1} ; u ; v ; \pi_C
\end{aligned}
$$

Where $\pi_B ; \pi_B^{-1} = \mathrm{id}_B$ follows from unpacking the definition of π and π^{-1}.

□

Corollary 7.23. *Let $t : A \to B$ be a term of a nominal monoidal theory. Modulo the equations of fig. 7.7 and fig. 7.8, the support of t is $A \cup B$.*

Proof. It follows from the proposition that $\mathbf{supp}\, t \subseteq A \cup B$. For the converse, suppose that there is $x \in A \cup B$ and a support S of t with $x \notin S \subseteq A \cup B$. Choose a permutation π that fixes S and maps x to some $\pi(x) \notin A \cup B$. Then either $\pi \cdot A \neq A$ or $\pi \cdot B \neq B$, hence $\pi \cdot t \neq t$, contradicting that S is a support of t.

□

The last corollary shows that internal names are bound by sequential composition. Indeed, in a composition $A \xrightarrow{t} C \xrightarrow{s} B$, the names in $C \backslash (A \cup B)$ do not appear in the support of $t ; s$.

7.4.3 Nominal PROPs

From the point of view of sec. 7.2, a nominal **PROP**, or nPROP for short, is an internal strict monoidal category in $(\mathsf{Nom}, 1, *)$ that has finite sets of names as objects and at least all bijections as arrows. A functor between nPROPs is an internal functor that preserves objects and bijections. We spell this out in detail.

Definition 7.24. An nPROP $(\mathbb{C}, \uplus, \mathbb{I})$ is a strict monoidal category internal in Nom, with a set $\mathbb{C}_0$ of 'objects' and a set $\mathbb{C}_1$ of 'arrows', defined as follows. We write **;** for the 'sequential' composition (in the diagrammatic order) and $\uplus$ for the 'parallel' or 'monoidal' composition.

- $\mathbb{C}$ is a category internal in $\underset{\sim}{\mathrm{Nom}}$. That means:

  - $\mathbb{C}_0$ is the (nominal) set of finite subsets of names $\mathcal{N}$. The support of a set of names A is the set itself: $\mathbf{supp}\,A = A$. The permutation action is given by $\pi \cdot A = \pi[A] = \{\pi(a) \mid a \in A\}$.

  - $\mathbb{C}_1$ contains at least all bijections ('renamings') $\pi_A : A \to \pi \cdot A$ for all finite permutations $\pi : \mathcal{N} \to \mathcal{N}$ and is closed under the operation mapping an arrow $f : A \to B$ to $\pi \cdot f : \pi \cdot A \to \pi \cdot B$ defined as $\pi \cdot f = (\pi_A)^{-1};f;\pi_B$. Such functions are referred to as finitely supported functions.

  - $\mathbf{dom}, \mathbf{cod}$ have the obvious definitions, taking $f : A \to B$ to A and B respectively and $i(A) = \mathrm{id}_A$.

- $\mathbb{C} \circledast \mathbb{C}$ is the separated-product-category internal in $\underset{\sim}{\mathrm{Nom}}$, where:

  - $(\mathbb{C} \circledast \mathbb{C})_0$ is the separated product on objects $\mathbb{C}_0 * \mathbb{C}_0 = \{(A, B) \in C_0 \times C_0 \mid \mathbf{supp}\,A \cap \mathbf{supp}\,B = \varnothing\}$. The permutation action is given by $\pi \cdot (A, B) = (\pi \cdot A, \pi \cdot B)$.
  - $(\mathbb{C} \circledast \mathbb{C})_1 = \{(f, g) \in C_1 \times C_1 \mid \mathbf{supp}(\mathbf{dom}\,f) \cap \mathbf{supp}(\mathbf{dom}\,g) = \varnothing = \mathbf{supp}(\mathbf{cod}\,f) \cap \mathbf{supp}(\mathbf{cod}\,g)\}$ is the subset of the cartesian product $\mathbb{C}_1 \times \mathbb{C}_1$ containing functions with disjoint support for their domains and co-domains.
  - $\mathbf{dom}, \mathbf{cod}$ are defined point-wise, i.e. $\mathbf{dom}(f, g) = (\mathbf{dom}\,f, \mathbf{dom}\,g)$ and $i(A, B) = (\mathrm{id}_A, \mathrm{id}_B)$.

- $\uplus : \mathbb{C} \circledast \mathbb{C} \to \mathbb{C}$ is an internal functor in $\underset{\sim}{\mathrm{Nom}}$, defined on objects $A \uplus B = A \cup B$ and arrows $f \uplus_1 g = f \cup g$.

- $\mathbb{I} : C_0$ is the unit to $\uplus$ and is defined as the empty set $\varnothing$.

Proof. To show that $\mathbb{C}$ is indeed an internal category, we have to check it satisfies def. B.1. However, we first further define:

- $\mathbb{C}_2$ as $\{(f, g) \mid \mathbf{cod}\,f = \mathbf{dom}\,g\}$ and π_1, π_2 as the obvious projections (these are different from the permutation action π defined earlier).
- $\mathbf{comp} : \mathbb{C}_2 \to \mathbb{C}_1$ is the usual function composition $\mathbf{comp}(f, g) = f\,;\,g$ (usually written as $g \circ f$).
- $\mathbb{C}_3 = \{(f, g, h) \mid \mathbf{cod}\,f = \mathbf{dom}\,g \wedge \mathbf{cod}\,g = \mathbf{dom}\,h\}$ where $\mathbf{left}(f, g, h) = (f, g)$ and $\mathbf{right}(f, g, h) = (g, h)$.
- $\mathbf{compl}, \mathbf{compr} : \mathbb{C}_3 \to \mathbb{C}_2$ where $\mathbf{compl}(f, g, h) = (f\,;\,g, h)$ and $\mathbf{compr}(f, g, h) = (f, g\,;\,h)$

Then, according to def. B.1, we need to show:

1) Since $\mathbb{C}_2$ is defined as a pullback, the diagram
$$\begin{tikzcd} \mathbb{C}_2 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & \mathbb{C}_1 \arrow[d, "\mathbf{dom}"] \\ \mathbb{C}_1 \arrow[r, "\mathbf{cod}"] & \mathbb{C}_0 \end{tikzcd}$$
commutes by definition.

2) For any $(f, g) \in \mathbb{C}_2$ we have

$$\mathbf{dom} \circ \mathbf{comp}(f, g) = \mathbf{dom}(f \ ; \ g) = \mathbf{dom}\, f = \mathbf{dom}(\pi_1(f, g)) = \mathbf{dom} \circ \pi_1(f, g)$$

and showing $\mathbf{cod} \circ \mathbf{comp} = \mathbf{cod} \circ \pi_2$ follows in exactly the same manner.

3) We have for all $A \in \mathbb{C}_0$,

$$\mathbf{dom} \circ i(A) = \mathbf{dom}(i(A)) = \mathbf{dom}\,\mathrm{id}_A = A = \mathrm{id}_{A_0}\, A = \mathbf{cod}\,\mathrm{id}_a = \mathbf{cod}(i(A)) = \mathbf{cod} \circ i(A)$$

4) For all $f : A \to B \in \mathbb{C}_1$ we have

$$\mathbf{comp} \circ \langle i \circ \mathbf{dom}, \mathrm{id}_{\mathbb{C}_1} \rangle(f) = \mathbf{comp}(i \circ \mathbf{dom}(f), \mathrm{id}_{\mathbb{C}_1}\, f) = \mathbf{comp}(i(A), f)$$
$$= \mathbf{comp}(\mathrm{id}_A, f) = \mathrm{id}_A \ ; \ f = f = \mathrm{id}_{A_1}\, f = f \ ; \ \mathrm{id}_B = \mathbf{comp}(f, \mathrm{id}_B) =$$
$$\mathbf{comp}(f, i(B)) = \mathbf{comp}(\mathrm{id}_{\mathbb{C}_1}\, f, i \circ \mathbf{cod}(f)) = \mathbf{comp} \circ \langle \mathrm{id}_{\mathbb{C}_1}, i \circ \mathbf{cod} \rangle(f)$$

5) Finally, we have

$$\mathbf{comp} \ \circ \ \mathbf{compl}(f, g, h) = \mathbf{comp}(\mathbf{compl}(f, g, h)) = \mathbf{comp}(f \ ; \ g, h)$$
$$= (f \ ; \ g) \ ; \ h = f \ ; \ (g \ ; \ h) =$$
$$\mathbf{comp}(f, g \ ; \ h) = \mathbf{comp}(\mathbf{compr}(f, g, h)) = \mathbf{comp} \ \circ \ \mathbf{compr}(f, g, h)$$

Next, rather than check that $\mathbb{C} \circledast \mathbb{C}$ is also an internal category, we can refer back to prop. 7.8, which tells us that given an arrow $\uplus : \mathbb{C}_0 * \mathbb{C}_0 \to \mathbb{C}_0$, we get a lifted $\uplus : \mathbb{C} \circledast \mathbb{C} \to \mathbb{C}$, such that $\mathbb{C} \circledast \mathbb{C}$ is an internal category and $\uplus$ is an internal functor. All we need to check is that our definitions of $(\mathbb{C} \circledast \mathbb{C})_1$ and $\uplus_1$ make the following limit diagram commute:

$$\begin{tikzcd} (\mathbb{C} \circledast \mathbb{C})_1 \arrow[rr, "\uplus_1"] \arrow[d, shift right, "\mathbf{dom}"'] \arrow[d, shift left, "\mathbf{cod}"] & & \mathbb{C}_1 \arrow[d, shift right, "\mathbf{dom}"'] \arrow[d, shift left, "\mathbf{cod}"] \\ (\mathbb{C} \circledast \mathbb{C})_0 \arrow[rr, "\uplus"'] & & \mathbb{C}_0 \end{tikzcd}$$

That is, for any $f : A \to B$ and $g : C \to D$ where $\mathbf{supp}(\mathbf{dom}\, f) \cap \mathbf{supp}(\mathbf{dom}\, g) = \mathbf{supp}(\mathbf{cod}\, f) \cap \mathbf{supp}(\mathbf{cod}\, g) = \varnothing$, we have

$$\uplus \circ \mathbf{dom}(f, g) = \mathbf{dom}\, f \uplus \mathbf{dom}\, g = A \uplus C = A \cup C = \mathbf{dom}(f \cup g) = \mathbf{dom}(f \uplus_1 g) = \mathbf{dom} \circ \uplus_1(f, g)$$

The case for **cod** follows in exactly the same manner.

Finally, we have to show that $\uplus$ is associative

$$f \uplus (g \uplus h) = f \cup (g \cup h) = (f \cup g) \cup h = (f \uplus g) \uplus h$$

and has $\mathbb{I}$ as its left and right identity

$$\mathbb{I} \uplus f = \varnothing \cup f = f = f \cup \varnothing = f \uplus \mathbb{I}$$

□

From this definition on can deduce the following.

Remark 7.25.

- A nominal PROP has a nominal set of objects and a nominal set of arrows.
- The support of an object A is A and the support of an arrow $f : A \to B$ is $A \cup B$. In particular, $\mathbf{supp}\,(f;g) = \mathbf{dom}\, f \cup \mathbf{cod}\, g$. In other words, nominal PROPs have diagrammatic α equivalence.
- There is a category nPROP that consists of nominal PROPs together with functors that are the identity on objects and strict monoidal and equivariant.
- Every NMT presents a nPROP. Conversely, every nPROP is presented by at least one NMT given by all terms as generators and all equations as equations.

7.5 Equivalence of nominal and ordinary PROPs

We show that the categories nPROP and PROP are equivalent.

To define translations between ordinary and nominal monoidal theories we introduce some auxiliary notation. We denote lists that contain each letter at most once by bold letters. If $\boldsymbol{a} = [a_1, \dots a_n]$ is a list, then $\underline{\boldsymbol{a}} = \{a_1, \dots a_n\}$. Given lists $\boldsymbol{a}$ and $\boldsymbol{a}'$ with $\underline{\boldsymbol{a}} = \underline{\boldsymbol{a}'}$ we abbreviate bijections in PROP (also called symmetries), mapping $i \mapsto j$ whenever $a_i = a_j'$, as $\langle \boldsymbol{a} | \boldsymbol{a}' \rangle$. Given lists $\boldsymbol{a}$ and $\boldsymbol{b}$ of the same length we write $[\boldsymbol{a}|\boldsymbol{b}] = \biguplus \delta_{a_i b_i}$ for the bijection $a_i \mapsto b_i$ in an nPROP.

Proposition 7.26. For any PROP $\mathcal{S}$, there is an nPROP

$$\mathsf{NOM}(\mathcal{S})$$

that has for all arrows $f : \underline{n} \to \underline{m}$ of $\mathcal{S}$, and for all lists $\boldsymbol{a} = [a_1, \dots a_n]$ and $\boldsymbol{b} = [b_1, \dots b_m]$

arrows $[\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \in \underset{\sim}{\mathrm{NOM}}(\mathcal{S})$. These arrows are subject to the following equations:

$$[\boldsymbol{a}\rangle f \,;\, g\langle \boldsymbol{c}] = [\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \,;\, [\boldsymbol{b}\rangle g\langle \boldsymbol{c}] \tag{NOM-1}$$

$$[\boldsymbol{a}+\!\!+\boldsymbol{c}\rangle f \oplus g\langle \boldsymbol{b}+\!\!+\boldsymbol{d}] = [\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle g\langle \boldsymbol{d}] \tag{NOM-2}$$

$$[\boldsymbol{a}\rangle \mathrm{id}\langle \boldsymbol{b}] = [\boldsymbol{a}|\boldsymbol{b}] \tag{NOM-3}$$

$$[\boldsymbol{a}\rangle \langle \boldsymbol{b}|\boldsymbol{b}'\rangle \,;\, f\langle \boldsymbol{c}] = [\boldsymbol{a}|\boldsymbol{b}] \,;\, [\boldsymbol{b}'\rangle f\langle \boldsymbol{c}] \tag{NOM-4}$$

$$[\boldsymbol{a}\rangle f \,;\, \langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{c}] = [\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \,;\, [\boldsymbol{b}'|\boldsymbol{c}] \tag{NOM-5}$$

Proof. To show that $\underset{\sim}{\mathrm{NOM}}(\mathcal{S})$ is well-defined, we need to check that the equations of $\mathcal{S}$ are respected. We only have space here for the most interesting case which is the naturality of symmetries given by the last equation in fig. 7.2. We write $\boldsymbol{a}^m$ for a list of a's of length m.

$$\begin{aligned}
&[\boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z\rangle (t \oplus id_z) \,;\, \sigma_{n,z}\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] \\
&= ([\boldsymbol{a}^m\rangle t\langle \boldsymbol{x}^n] \uplus [\boldsymbol{a}^z\rangle id_z\langle \boldsymbol{x}^z]) \,;\, [\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z\rangle \sigma_{n,z}\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(NOM-1,2)}\\
&= ([\boldsymbol{a}^z\rangle id_z\langle \boldsymbol{x}^z] \uplus [\boldsymbol{a}^m\rangle t\langle \boldsymbol{x}^n]) \,;\, [\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z\rangle \sigma_{n,z}\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(NMT-comm)}\\
&= [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n] \,;\, [\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z\rangle \sigma_{n,z}\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(NOM-2)}\\
&= [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n] \,;\, [\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z\rangle \langle \boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z|\boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n\rangle\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && (\sigma\text{-def})\\
&= [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n] \,;\, [\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z|\boldsymbol{x}^n +\!\!+ \boldsymbol{x}^z] \,;\, [\boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n|\boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(41-2)}\\
&= [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n] \,;\, [\boldsymbol{x}^z +\!\!+ \boldsymbol{x}^n|\boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && (\delta_{aa} = id_a)\\
&= [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(NOM-5)}\\
&= [\boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z|\boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z] \,;\, [\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle id_z \oplus t\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && (\delta_{aa} = id_a)\\
&= [\boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z\rangle \langle \boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z|\boldsymbol{a}^z +\!\!+ \boldsymbol{a}^m\rangle \,;\, (id_z \oplus t)\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && \text{(NOM-4)}\\
&= [\boldsymbol{a}^m +\!\!+ \boldsymbol{a}^z\rangle \sigma_{m,z} \,;\, (id_z \oplus t)\langle \boldsymbol{b}^z +\!\!+ \boldsymbol{b}^n] && (\sigma\text{-def})
\end{aligned}$$

Note how commutativity of $\uplus$ is used to show that naturality of symmetries is respected. □

Example 7.27. $\underset{\sim}{\mathrm{n}\mathbb{F}}$ is isomorphic to $\underset{\sim}{\mathrm{NOM}}(\mathbb{F})$.

Proof. We define a map $G : \underset{\sim}{\mathrm{NOM}}(\mathbb{F}) \to \underset{\sim}{\mathrm{n}\mathbb{F}}$ as

$$G([\boldsymbol{a}\rangle f\langle \boldsymbol{b}]) = [\![\boldsymbol{a}\rangle\!\rangle f\langle\!\langle \boldsymbol{b}]\!] \text{ where } f : n \to m$$

The semantic brackets $[\![-\rangle\!\rangle - \langle\!\langle -]\!]$ translate the arrow $f \in \mathbb{F}$ into an arrow in $\underset{\sim}{\mathrm{n}\mathbb{F}}$ by pre-composing with $\vec{\boldsymbol{a}} : A \to n$ and post-composing with $\vec{\boldsymbol{b}}^{-1} : m \to B$, where $\vec{\boldsymbol{a}}$ is a bijection between the underlying set of $\boldsymbol{a}$ and n, given by the ordering of the list $\boldsymbol{a}$. Therefore, we have $[\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!] \overset{\text{def}}{=} \vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}$.

G is defined on the free nPROP generated by $\{[\boldsymbol{a}\rangle f \langle \boldsymbol{b}] \mid f \in \mathbb{F}\}$. In particular, G is a homomorphism:

$$G([\boldsymbol{a}|\boldsymbol{b}]) = [\boldsymbol{a}|\boldsymbol{b}]$$

$$G(f \ ; \ g) = G(f) \ ; \ G(g)$$

$$G(f \uplus g) = G(f) \uplus G(g)$$

We show G is well defined, that is, it respects the equations of NOM, namely $f = g$ in NOM($\mathbb{F}$) implies $G(f) = G(g)$ in n$\mathbb{F}$:

$$\begin{aligned}
G([\boldsymbol{a}\rangle f \ ; \ g\langle \boldsymbol{c}]) &= [\![\boldsymbol{a}\rangle\!\rangle f \ ; \ g\langle\!\langle \boldsymbol{c}]\!] \\
&= \vec{\boldsymbol{a}} \ ; \ f \ ; \ g \ ; \ \vec{\boldsymbol{c}}^{-1} \\
&= \vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1} \ ; \ \vec{\boldsymbol{b}} \ ; \ g \ ; \ \vec{\boldsymbol{c}}^{-1} \\
&= [\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!] \ ; \ [\![\boldsymbol{b}\rangle\!\rangle g \langle\!\langle \boldsymbol{c}]\!] \\
&= G([\boldsymbol{a}\rangle f \langle \boldsymbol{b}]) \ ; \ G([\boldsymbol{b}\rangle g \langle \boldsymbol{c}]) \\
&= G([\boldsymbol{a}\rangle f \langle \boldsymbol{b}] \ ; \ [\boldsymbol{b}\rangle g \langle \boldsymbol{c}])
\end{aligned}$$

$$\begin{aligned}
G([\boldsymbol{a} + \boldsymbol{c}\rangle f \oplus g \langle \boldsymbol{b} + \boldsymbol{d}]) &= [\![\boldsymbol{a} + \boldsymbol{c}\rangle\!\rangle f \oplus g \langle\!\langle \boldsymbol{b} + \boldsymbol{d}]\!] \\
&= \overrightarrow{\boldsymbol{a} + \boldsymbol{c}} \ ; \ (f \oplus g) \ ; \ \overrightarrow{\boldsymbol{b} + \boldsymbol{d}}^{-1} \\
&= (\vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}) \uplus (\vec{\boldsymbol{c}} \ ; \ g \ ; \ \vec{\boldsymbol{d}}^{-1}) \\
&= [\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!] \uplus [\![\boldsymbol{c}\rangle\!\rangle g \langle\!\langle \boldsymbol{d}]\!] \\
&= G([\boldsymbol{a}\rangle f \langle \boldsymbol{b}]) \uplus G([\boldsymbol{c}\rangle g \langle \boldsymbol{d}]) \\
&= G([\boldsymbol{a}\rangle f \langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle g \langle \boldsymbol{d}])
\end{aligned}$$

We justify the third equation in the above proof

$$\overrightarrow{\boldsymbol{a} + \boldsymbol{c}} \ ; \ f \oplus g \ ; \ \overrightarrow{\boldsymbol{b} + \boldsymbol{d}}^{-1} = (\vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}) \uplus (\vec{\boldsymbol{c}} \ ; \ g \ ; \ \vec{\boldsymbol{d}}^{-1})$$

where $f : n \to m$, $g : o \to p$ and $|\boldsymbol{a}| = n$, $|\boldsymbol{c}| = o$, $|\boldsymbol{b}| = m$, $|\boldsymbol{d}| = p$, with the following argument. Recall that $f \uplus g$ is just the set union of the two functions $f \cup g$ and that

$f \oplus g$ is defined for $f : n \to m$ and $g : o \to p$ in the following way:

$$f \oplus g\,(k) = \begin{cases} f(k) & \text{if } k \leq n \\ g(k - n) + m & \text{otherwise} \end{cases}$$

Its also easy enough to see that we have:

$$\boldsymbol{a} \vec{+\!\!+} \boldsymbol{b}\,(x) = \begin{cases} \vec{\boldsymbol{a}}(x) & \text{if } x \in set(\boldsymbol{a}) \\ \vec{\boldsymbol{b}}(x) + |\boldsymbol{a}| & \text{otherwise} \end{cases} \qquad \boldsymbol{a} \vec{+\!\!+} \boldsymbol{b}^{-1}(k) = \begin{cases} \vec{\boldsymbol{a}}^{-1}(k) & \text{if } k \leq |\boldsymbol{a}| \\ \vec{\boldsymbol{b}}^{-1}(k - |\boldsymbol{a}|) & \text{otherwise} \end{cases}$$

Then, given an x we have two cases:

- $x \in set(\boldsymbol{a})$. Then we have:

$$\begin{aligned} \boldsymbol{a} \vec{+\!\!+} \boldsymbol{c} \;;\; f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(x) &= f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(\boldsymbol{a} \vec{+\!\!+} \boldsymbol{c}\,(x)) \\ &= f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(\vec{\boldsymbol{a}}(x)) \\ &= \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(f \oplus g\,(\vec{\boldsymbol{a}}(x))) \\ &= \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(f(\vec{\boldsymbol{a}}(x))) \\ &= \vec{\boldsymbol{b}}^{-1}(f(\vec{\boldsymbol{a}}(x))) \\ &= \vec{\boldsymbol{a}} \;;\; f \;;\; \vec{\boldsymbol{b}}^{-1}(x) \end{aligned}$$

- $x \in set(\boldsymbol{c})$. Then we have:

$$\begin{aligned} \boldsymbol{a} \vec{+\!\!+} \boldsymbol{c} \;;\; f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(x) &= f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(\boldsymbol{a} \vec{+\!\!+} \boldsymbol{c}\,(x)) \\ &= f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(\vec{\boldsymbol{c}}(x) + |\boldsymbol{a}|) \\ &= \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(f \oplus g\,(\vec{\boldsymbol{c}}(x) + |\boldsymbol{a}|)) \\ &= \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(g(\vec{\boldsymbol{c}}(x) + |\boldsymbol{a}| - |\boldsymbol{a}|) + |\boldsymbol{b}|) \\ &= \vec{\boldsymbol{d}}^{-1}(g(\vec{\boldsymbol{c}}(x)) + |\boldsymbol{b}| - |\boldsymbol{b}|) \\ &= \vec{\boldsymbol{c}} \;;\; f \;;\; \vec{\boldsymbol{d}}^{-1}(x) \end{aligned}$$

Thus we can conclude that

$$\boldsymbol{a} \vec{+\!\!+} \boldsymbol{c} \;;\; f \oplus g \;;\; \boldsymbol{b} \vec{+\!\!+} \boldsymbol{d}^{-1}(x) = (\vec{\boldsymbol{a}} \;;\; f \;;\; \vec{\boldsymbol{b}}^{-1}) \cup (\vec{\boldsymbol{c}} \;;\; g \;;\; \vec{\boldsymbol{d}}^{-1})(x)$$

for any x, from which the original proposition follows.

Having shown

$$G([\boldsymbol{a} +\!\!+ \boldsymbol{c}\rangle f \oplus g\langle \boldsymbol{b} +\!\!+ \boldsymbol{d}]) = [\![\boldsymbol{a} +\!\!+ \boldsymbol{c}\rangle\!\rangle f \oplus g\langle\!\langle \boldsymbol{b} +\!\!+ \boldsymbol{d}]\!] = G([\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle g\langle \boldsymbol{d}])$$

we have to show that the last three equations in $\underset{\sim}{\mathrm{NOM}}$ are respected by $\boldsymbol{G}$:

$$G([\boldsymbol{a}\rangle id\langle\boldsymbol{b}]) = [\![\boldsymbol{a}\rangle\!\rangle id\langle\!\langle\boldsymbol{b}]\!] = \vec{\boldsymbol{a}}\ ;\ id\ ;\ \vec{\boldsymbol{b}}^{-1} = [\boldsymbol{a}|\boldsymbol{b}] = G([\boldsymbol{a}|\boldsymbol{b}])$$

$$\begin{aligned} G([\boldsymbol{a}\rangle\langle\boldsymbol{b}|\boldsymbol{b}'\rangle\ ;\ f\langle\boldsymbol{c}]) &= [\![\boldsymbol{a}\rangle\!\rangle\langle\boldsymbol{b}|\boldsymbol{b}'\rangle\ ;\ f\langle\!\langle\boldsymbol{c}]\!] \\ &= \vec{\boldsymbol{a}}\ ;\ \langle\boldsymbol{b}|\boldsymbol{b}'\rangle\ ;\ f\ ;\ \vec{\boldsymbol{c}}^{-1} \\ &= \vec{\boldsymbol{a}}\ ;\ \vec{\boldsymbol{b}}^{-1}\ ;\ \vec{\boldsymbol{b}}'\ ;\ f\ ;\ \vec{\boldsymbol{c}}^{-1} \\ &= [\boldsymbol{a}|\boldsymbol{b}]\ ;\ \vec{\boldsymbol{b}}'\ ;\ f\ ;\ \vec{\boldsymbol{c}}^{-1} \\ &= [\boldsymbol{a}|\boldsymbol{b}]\ ;\ [\![\boldsymbol{b}'\rangle\!\rangle f\langle\!\langle\boldsymbol{c}]\!] \\ &= G([\boldsymbol{a}|\boldsymbol{b}])\ ;\ G([\boldsymbol{b}'\rangle f\langle\boldsymbol{c}]) \\ &= G([\boldsymbol{a}|\boldsymbol{b}]\ ;\ [\boldsymbol{b}'\rangle f\langle\boldsymbol{c}]) \end{aligned}$$

This concludes the proof that $\boldsymbol{G}$ is a well defined function.

Now we define the map $H : \underset{\sim}{\mathrm{n}\mathbb{F}} \to \underset{\sim}{\mathrm{NOM}(\mathbb{F})}$ as

$$H(f) = [\boldsymbol{a}\rangle\langle\!\langle\boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle\langle\boldsymbol{b}]$$

for $f : A \to B$ and $\boldsymbol{a}, \boldsymbol{b}$ s.t. $set(\boldsymbol{a}) = A$ and $set(\boldsymbol{b}) = B$.

Similarly to $\boldsymbol{G}$, the semantic brackets $\langle\!\langle -]\!] - [\![- \rangle\!\rangle$ translate the arrow $\boldsymbol{f} \in \underset{\sim}{\mathrm{n}\mathbb{F}}$ into an arrow in $\underset{\sim}{\mathbb{F}}$, defined as $\langle\!\langle\boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle \overset{\mathrm{def}}{=} \vec{\boldsymbol{a}}^{-1}\ ;\ f\ ;\ \vec{\boldsymbol{b}}$.

We show that $\boldsymbol{H}$ is a homomorphism:

For $f : A \to B$ and $g : B \to C$, we have:

$$\begin{aligned} H(f\ ;\ g) &= [\boldsymbol{a}\rangle\langle\!\langle\boldsymbol{a}]\!] f\ ;\ g [\![\boldsymbol{c}\rangle\!\rangle\langle\boldsymbol{c}] \\ &= [\boldsymbol{a}\rangle\vec{\boldsymbol{a}}^{-1}\ ;\ f\ ;\ g\ ;\ \vec{\boldsymbol{c}}\langle\boldsymbol{c}] \\ &= [\boldsymbol{a}\rangle\vec{\boldsymbol{a}}^{-1}\ ;\ f\ ;\ \vec{\boldsymbol{b}}\ ;\ \vec{\boldsymbol{b}}^{-1}\ ;\ g\ ;\ \vec{\boldsymbol{c}}\langle\boldsymbol{c}] \\ &= [\boldsymbol{a}\rangle\vec{\boldsymbol{a}}^{-1}\ ;\ f\ ;\ \vec{\boldsymbol{b}}\langle\boldsymbol{b}]\ ;\ [\boldsymbol{b}\rangle\vec{\boldsymbol{b}}^{-1}\ ;\ g\ ;\ \vec{\boldsymbol{c}}\langle\boldsymbol{c}] \\ &= [\boldsymbol{a}\rangle\langle\!\langle\boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle\langle\boldsymbol{b}]\ ;\ [\boldsymbol{b}\rangle\langle\!\langle\boldsymbol{b}]\!] g [\![\boldsymbol{c}\rangle\!\rangle\langle\boldsymbol{c}] \\ &= H(f)\ ;\ H(g) \end{aligned}$$

For $f : A \to B$, $g : C \to D$, $X = A \cup C$ and $Y = B \cup D$ we have:

$$
\begin{aligned}
H(f \uplus g) &= [\mathbf{x}\rangle\langle\!\langle \mathbf{x}]\!] f \uplus g [\![\mathbf{y}\rangle\!\rangle\langle \mathbf{y}] \\
&= [\mathbf{x}\rangle \vec{\mathbf{x}}^{-1} \; ; \; (f \uplus g) \; ; \; \vec{\mathbf{y}}\langle \mathbf{y}] \\
&= [\mathbf{x}\rangle \vec{\mathbf{x}}^{-1} \; ; \; (\vec{\boldsymbol{a}} \uplus \vec{\boldsymbol{c}}) \; ; \; (\vec{\boldsymbol{a}}^{-1} \uplus \vec{\boldsymbol{c}}^{-1}) \; ; \; (f \uplus g) \; ; \; (\vec{\boldsymbol{b}} \uplus \vec{\boldsymbol{d}}) \; ; \; (\vec{\boldsymbol{b}}^{-1} \uplus \vec{\boldsymbol{d}}^{-1}) \; ; \; \vec{\mathbf{y}}\langle \mathbf{y}] \\
&= [\mathbf{x}\rangle \vec{\mathbf{x}}^{-1} \; ; \; (\vec{\boldsymbol{a}} \uplus \vec{\boldsymbol{c}}) \; ; \; ((\vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}}) \uplus (\vec{\boldsymbol{c}}^{-1} \; ; \; g \; ; \; \vec{\boldsymbol{d}})) \; ; \; (\vec{\boldsymbol{b}}^{-1} \uplus \vec{\boldsymbol{d}}^{-1}) \; ; \; \vec{\mathbf{y}}\langle \mathbf{y}] \\
&= [\mathbf{x}\rangle\langle \mathbf{x} | \boldsymbol{a} + \boldsymbol{c}\rangle \; ; \; ((\vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}}) \uplus (\vec{\boldsymbol{c}}^{-1} \; ; \; g \; ; \; \vec{\boldsymbol{d}})) \; ; \; \langle \boldsymbol{b} + \boldsymbol{d} | \mathbf{y}\rangle\langle \mathbf{y}] \\
&= [\boldsymbol{a} + \boldsymbol{c}\rangle(\vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}}) \uplus (\vec{\boldsymbol{c}}^{-1} \; ; \; g \; ; \; \vec{\boldsymbol{d}})\langle \boldsymbol{b} + \boldsymbol{d}] \\
&= [\boldsymbol{a} + \boldsymbol{c}\rangle\langle\!\langle \boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle \uplus \langle\!\langle \boldsymbol{c}]\!] g [\![\boldsymbol{d}\rangle\!\rangle\langle \boldsymbol{b} + \boldsymbol{d}] \\
&= [\boldsymbol{a}\rangle\langle\!\langle \boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle\langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle\langle\!\langle \boldsymbol{c}]\!] g [\![\boldsymbol{d}\rangle\!\rangle\langle \boldsymbol{d}] \\
&= H(f) \uplus H(g)
\end{aligned}
$$

$$
\begin{aligned}
H([\boldsymbol{a}|\boldsymbol{b}]) &= [\boldsymbol{a}'\rangle\langle\!\langle \boldsymbol{a}']\!][\boldsymbol{a}|\boldsymbol{b}][\![\boldsymbol{b}'\rangle\!\rangle\langle \boldsymbol{b}'] \\
&= [\boldsymbol{a}'\rangle \, \vec{\boldsymbol{a}'}^{-1} \; ; \; [\boldsymbol{a}|\boldsymbol{b}] \; ; \; \vec{\boldsymbol{b}'} \, \langle \boldsymbol{b}'] \\
&= [\boldsymbol{a}'\rangle \, \vec{\boldsymbol{a}'}^{-1} \; ; \; \vec{\boldsymbol{a}} \; ; \; \vec{\boldsymbol{b}}^{-1} \; ; \; \vec{\boldsymbol{b}'} \, \langle \boldsymbol{b}'] \\
&= [\boldsymbol{a}'\rangle\langle \boldsymbol{a}' | \boldsymbol{a}\rangle \; ; \; \langle \boldsymbol{b} | \boldsymbol{b}'\rangle\langle \boldsymbol{b}'] \\
&= [\boldsymbol{a}\rangle id \langle \boldsymbol{b}] = [\boldsymbol{a}|\boldsymbol{b}]
\end{aligned}
$$

Having shown H is a homomorphism, we finally show that G and H are isomorphisms.

$$
G \circ H(f) = G([\boldsymbol{a}\rangle\langle\!\langle \boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle\langle \boldsymbol{b}]) = [\![\boldsymbol{a}\rangle\!\rangle\langle\!\langle \boldsymbol{a}]\!] f [\![\boldsymbol{b}\rangle\!\rangle\langle\!\langle \boldsymbol{b}]\!] = \vec{\boldsymbol{a}} \; ; \; \vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}}^{-1} \; ; \; \vec{\boldsymbol{b}} = f
$$

$$
\begin{aligned}
H \circ G([\boldsymbol{a}\rangle f \langle \boldsymbol{b}]) &= H([\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!]) \\
&= [\boldsymbol{a}'\rangle\langle\!\langle \boldsymbol{a}']\!][\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!][\![\boldsymbol{b}'\rangle\!\rangle\langle \boldsymbol{b}'] \\
&= [\boldsymbol{a}'\rangle\langle \boldsymbol{a}' | \boldsymbol{a}\rangle \; ; \; f \; ; \; \langle \boldsymbol{b} | \boldsymbol{b}'\rangle\langle \boldsymbol{b}'] = [\boldsymbol{a}\rangle f \langle \boldsymbol{b}]
\end{aligned}
$$

□

Remark 7.28. The argument in the example above can easily be adapted to the categories of injections $n\mathbb{I}$, surjections $n\mathbb{S}$, bijections $n\mathbb{B}$, partial functions $n\mathbb{P}$ and relations $n\mathbb{R}$, being isomorphic to NOM($\mathbb{I}$), NOM($\mathbb{S}$), NOM($\mathbb{B}$), etc. The same construction works in all the other instances, since the equational reasoning above only requires bijective arrows/functions to be present in a given category.

Proposition 7.29. For any nPROP $\mathcal{T}$ there is a PROP

$$\mathrm{ORD}(\mathcal{T})$$

that has for all arrows $f : A \to B$ of $\mathcal{T}$, and for all lists $\boldsymbol{a} = [a_1, \dots a_n]$ and $\boldsymbol{b} = [b_1, \dots b_m]$ arrows $\langle \boldsymbol{a}] f [\boldsymbol{b}\rangle$. These arrows are subject to the equations below:

$$\langle \boldsymbol{a}]\, f \;;\; g\, [\boldsymbol{c}\rangle = \langle \boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{b}]\, g\, [\boldsymbol{c}\rangle \tag{ORD-1}$$

$$\langle \boldsymbol{a}_f + \boldsymbol{a}_g]\, f \uplus g\, [\boldsymbol{b}_f + \boldsymbol{b}_g\rangle = \langle \boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle \oplus \langle \boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle \tag{ORD-2}$$

$$\langle \boldsymbol{a}]\, \mathrm{id}\, [\boldsymbol{a}\rangle = \mathrm{id} \tag{ORD-3}$$

$$\langle \boldsymbol{a}]\, [\boldsymbol{a}' | \boldsymbol{b}] \;;\; f\, [\boldsymbol{c}\rangle = \langle \boldsymbol{a} | \boldsymbol{a}'\rangle \;;\; \langle \boldsymbol{b}]\, f\, [\boldsymbol{c}\rangle \tag{ORD-4}$$

$$\langle \boldsymbol{a}]\, f \;;\; [\boldsymbol{b} | \boldsymbol{c}]\, [\boldsymbol{c}'\rangle = \langle \boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{c} | \boldsymbol{c}'\rangle \tag{ORD-5}$$

$$\langle \boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle = \langle \pi \cdot \boldsymbol{a}]\, \pi \cdot f\, [\pi \cdot \boldsymbol{b}\rangle \tag{ORD-6}$$

Proof. To show that ORD is well-defined we need to show that the equations **NMT** are respected. The most interesting case here is the commutativity of $\uplus$ since the $\oplus$ of SMTs is not commutative.

$$\begin{aligned}
&\langle \boldsymbol{a}_t + \boldsymbol{a}_s]\, t \uplus s\, [\boldsymbol{b}_t + \boldsymbol{b}_s\rangle \\
&= \langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle \oplus \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle && \text{(ORD-2)} \\
&= (\langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle \;;\; id_{|\boldsymbol{b}_t|}) \oplus (id_{|\boldsymbol{a}_s|} \;;\; \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle) && (id \;;\; a = a = a \;;\; id) \\
&= (\langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle \oplus id_{|\boldsymbol{a}_s|}) \;;\; (id_{|\boldsymbol{b}_t|} \oplus \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle) && \text{(SMT-ch)} \\
&= (\langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle \oplus id_{|\boldsymbol{a}_s|}) \;;\; \sigma_{|\boldsymbol{b}_t|,|\boldsymbol{a}_s|} \;;\; \sigma_{|\boldsymbol{a}_s|,|\boldsymbol{b}_t|} \;;\; (id_{|\boldsymbol{b}_t|} \oplus \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle) && \text{(SMT-sym)} \\
&= \sigma_{|\boldsymbol{a}_t|,|\boldsymbol{a}_s|} \;;\; (id_{|\boldsymbol{a}_s|} \oplus \langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle) \;;\; \sigma_{|\boldsymbol{a}_s|,|\boldsymbol{b}_t|} \;;\; (id_{|\boldsymbol{b}_t|} \oplus \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle) && \text{(SMT-nat)} \\
&= \sigma_{|\boldsymbol{a}_t|,|\boldsymbol{a}_s|} \;;\; (id_{|\boldsymbol{a}_s|} \oplus \langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle) \;;\; (\langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle \oplus id_{|\boldsymbol{b}_t|}) \;;\; \sigma_{|\boldsymbol{b}_s|,|\boldsymbol{b}_t|} && \text{(SMT-nat)} \\
&= \sigma_{|\boldsymbol{a}_t|,|\boldsymbol{a}_s|} \;;\; ((id_{|\boldsymbol{a}_s|} \;;\; \langle \boldsymbol{a}_s]\, s\, [\boldsymbol{b}_s\rangle) \oplus (\langle \boldsymbol{a}_t]\, t\, [\boldsymbol{b}_t\rangle \;;\; id_{|\boldsymbol{b}_t|})) \;;\; \sigma_{|\boldsymbol{b}_s|,|\boldsymbol{b}_t|} && \text{(SMT-ch)} \\
&= \sigma_{|\boldsymbol{a}_t|,|\boldsymbol{a}_s|} \;;\; \langle \boldsymbol{a}_s + \boldsymbol{a}_t]\, s \uplus t\, [\boldsymbol{b}_s + \boldsymbol{b}_t\rangle \;;\; \sigma_{|\boldsymbol{b}_s|,|\boldsymbol{b}_t|} && (id \;;\; a = a, \text{ORD-2}) \\
&= \langle \boldsymbol{a}_t + \boldsymbol{a}_s | \boldsymbol{a}_s + \boldsymbol{a}_t\rangle \;;\; \langle \boldsymbol{a}_s + \boldsymbol{a}_t]\, s \uplus t\, [\boldsymbol{b}_s + \boldsymbol{b}_t\rangle \;;\; \langle \boldsymbol{b}_s + \boldsymbol{b}_t | \boldsymbol{b}_t + \boldsymbol{b}_s\rangle && (\sigma\text{-def}) \\
&= \langle \boldsymbol{a}_t + \boldsymbol{a}_s]\, [\boldsymbol{a}_s + \boldsymbol{a}_t | \boldsymbol{a}_s + \boldsymbol{a}_t] \;;\; s \uplus t \;;\; [\boldsymbol{b}_s + \boldsymbol{b}_t | \boldsymbol{b}_s + \boldsymbol{b}_t]\, [\boldsymbol{b}_t + \boldsymbol{b}_s\rangle && \text{(ORD-4,5)} \\
&= \langle \boldsymbol{a}_t + \boldsymbol{a}_s]\, s \uplus t\, [\boldsymbol{b}_t + \boldsymbol{b}_s\rangle && (\delta_{aa} = id_a)
\end{aligned}$$

Note how naturality of symmetries is used to show that the definition of ORD respects commutativity of $\uplus$.

□

Example 7.30. $\underset{\sim}{\mathbb{F}}$ is isomorphic to $\underset{\sim}{\mathrm{ORD}}(\mathrm{n}\underset{\sim}{\mathbb{F}})$.

Proof. We define a map $G : \underset{\sim}{\mathrm{ORD}}(\mathrm{n}\underset{\sim}{\mathbb{F}}) \to \underset{\sim}{\mathbb{F}}$ as

$$G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) = \langle\!\langle \boldsymbol{a}]\!]f[\![\boldsymbol{b}\rangle\!\rangle \text{ where } f : A \to B$$

G is defined on the free $\underset{\sim}{\mathrm{PROP}}$ generated by $\{\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle \mid f \in \mathrm{n}\underset{\sim}{\mathbb{F}}\}$. In particular, G is a homomorphism:

$$\begin{aligned}
G(\sigma) &= \sigma \\
G(f \; ; \; g) &= G(f) \; ; \; G(g) \\
G(f \oplus g) &= G(f) \oplus G(g)
\end{aligned}$$

We show G is well defined, that is, it respects the equations of $\underset{\sim}{\mathrm{ORD}}$:

$$\begin{aligned}
G(\langle \boldsymbol{a}]f \; ; \; g[\boldsymbol{c}\rangle) &= \langle\!\langle \boldsymbol{a}]\!]f \; ; \; g[\![\boldsymbol{c}\rangle\!\rangle \\
&= \vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; g \; ; \; \vec{\boldsymbol{c}} \\
&= \vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}} \; ; \; \vec{\boldsymbol{b}}^{-1} \; ; \; g \; ; \; \vec{\boldsymbol{c}} \\
&= \langle\!\langle \boldsymbol{a}]\!]f[\![\boldsymbol{b}\rangle\!\rangle \; ; \; \langle\!\langle \boldsymbol{b}]\!]g[\![\boldsymbol{c}\rangle\!\rangle \\
&= G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) \; ; \; G(\langle \boldsymbol{b}]g[\boldsymbol{c}\rangle) \\
&= G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle \; ; \; \langle \boldsymbol{b}]g[\boldsymbol{c}\rangle)
\end{aligned}$$

$$\begin{aligned}
G(\langle \boldsymbol{a} + \boldsymbol{c}]f \uplus g[\boldsymbol{b} + \boldsymbol{d}\rangle) &= [\![\boldsymbol{a} + \boldsymbol{c}\rangle\!\rangle f \uplus g\langle\!\langle \boldsymbol{b} + \boldsymbol{d}]\!] \\
&= \overrightarrow{\boldsymbol{a} + \boldsymbol{c}}^{-1} \; ; \; (f \uplus g) \; ; \; \overrightarrow{\boldsymbol{b} + \boldsymbol{d}} \\
&= (\vec{\boldsymbol{a}}^{-1} \; ; \; f \; ; \; \vec{\boldsymbol{b}}) \oplus (\vec{\boldsymbol{c}}^{-1} \; ; \; g \; ; \; \vec{\boldsymbol{d}}) \\
&= \langle\!\langle \boldsymbol{a}]\!]f[\![\boldsymbol{b}\rangle\!\rangle \oplus \langle\!\langle \boldsymbol{c}]\!]g[\![\boldsymbol{d}\rangle\!\rangle \\
&= G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) \oplus G(\langle \boldsymbol{c}]g[\boldsymbol{d}\rangle) \\
&= G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle \oplus \langle \boldsymbol{c}]g[\boldsymbol{d}\rangle)
\end{aligned}$$

$$G(\langle \boldsymbol{a}]id[\boldsymbol{a}\rangle) = \langle\!\langle \boldsymbol{a}]\!]id[\![\boldsymbol{a}\rangle\!\rangle = \vec{\boldsymbol{a}}^{-1} \; ; \; id \; ; \; \vec{\boldsymbol{a}} = id = G(id)$$

$$
\begin{aligned}
G(\langle \boldsymbol{a}][\boldsymbol{a}'|\boldsymbol{b}] \,;\, f[\boldsymbol{c}\rangle) &= \langle\!\langle \boldsymbol{a}\rrbracket[\boldsymbol{a}'|\boldsymbol{b}] \,;\, f\llbracket \boldsymbol{c}\rangle\!\rangle \\
&= \vec{\boldsymbol{a}}^{-1} \,;\, [\boldsymbol{a}'|\boldsymbol{b}] \,;\, f \,;\, \vec{\boldsymbol{c}} \\
&= \vec{\boldsymbol{a}}^{-1} \,;\, \vec{\boldsymbol{a}}' \,;\, \vec{\boldsymbol{b}}^{-1} \,;\, f \,;\, \vec{\boldsymbol{c}} \\
&= \langle \boldsymbol{a}|\boldsymbol{b}\rangle \,;\, \vec{\boldsymbol{b}}^{-1} \,;\, f \,;\, \vec{\boldsymbol{c}} \\
&= \langle \boldsymbol{a}|\boldsymbol{b}\rangle \,;\, \langle\!\langle \boldsymbol{b}\rrbracket f\llbracket \boldsymbol{c}\rangle\!\rangle \\
&= G(\langle \boldsymbol{a}|\boldsymbol{b}\rangle) \,;\, G(\langle \boldsymbol{b}]f[\boldsymbol{c}\rangle) \\
&= G(\langle \boldsymbol{a}|\boldsymbol{b}\rangle \,;\, \langle \boldsymbol{b}]f[\boldsymbol{c}\rangle)
\end{aligned}
$$

This concludes the proof that G is a well defined function.

Now we define the map $H : \mathrm{n}\mathbb{F} \to \mathrm{NOM}(\mathbb{F})$ as

$$H(f) = \langle \boldsymbol{a}]\llbracket \boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}\rrbracket[\boldsymbol{b}\rangle$$

showing that it is a homomorphism:

$$
\begin{aligned}
H(f \,;\, g) &= \langle \boldsymbol{a}]\llbracket \boldsymbol{a}\rangle\!\rangle f \,;\, g\langle\!\langle \boldsymbol{c}\rrbracket[\boldsymbol{c}\rangle \\
&= \langle \boldsymbol{a}]\vec{\boldsymbol{a}} \,;\, f \,;\, g \,;\, \vec{\boldsymbol{c}}^{-1}[\boldsymbol{c}\rangle \\
&= \langle \boldsymbol{a}]\vec{\boldsymbol{a}} \,;\, f \,;\, \vec{\boldsymbol{b}}^{-1} \,;\, \vec{\boldsymbol{b}} \,;\, g \,;\, \vec{\boldsymbol{c}}\langle \boldsymbol{c}] \\
&= \langle \boldsymbol{a}]\vec{\boldsymbol{a}} \,;\, f \,;\, \vec{\boldsymbol{b}}^{-1}[\boldsymbol{b}\rangle \,;\, \langle \boldsymbol{b}]\vec{\boldsymbol{b}} \,;\, g \,;\, \vec{\boldsymbol{c}}^{-1}[\boldsymbol{c}\rangle \\
&= \langle \boldsymbol{a}]\llbracket \boldsymbol{a}\rangle\!\rangle f\langle\!\langle \boldsymbol{b}\rrbracket[\boldsymbol{b}\rangle \,;\, \langle \boldsymbol{b}]\llbracket \boldsymbol{b}\rangle\!\rangle g\langle\!\langle \boldsymbol{c}\rrbracket[\boldsymbol{c}\rangle \\
&= H(f) \,;\, H(g)
\end{aligned}
$$

$$
\begin{aligned}
H(f \oplus g) &= \langle \mathbf{x}]\![\![\mathbf{x}\rangle\!\rangle f \oplus g \langle\!\langle \mathbf{y}]\!]\![\mathbf{y}\rangle \\
&= \langle \mathbf{x}]\vec{\mathbf{x}} \ ; \ (f \oplus g) \ ; \ \vec{\mathbf{y}}^{-1}[\mathbf{y}\rangle \\
&= \langle \mathbf{x}]\vec{\mathbf{x}} \ ; \ (\vec{\boldsymbol{a}}^{-1} \uplus \vec{\boldsymbol{c}}^{-1}) \ ; \ (\vec{\boldsymbol{a}} \uplus \vec{\boldsymbol{c}}) \ ; \ (f \oplus g) \ ; \ (\vec{\boldsymbol{b}}^{-1} \uplus \vec{\boldsymbol{d}}^{-1}) \ ; \ (\vec{\boldsymbol{b}} \uplus \vec{\boldsymbol{d}}) \ ; \ \vec{\mathbf{y}}^{-1}[\mathbf{y}\rangle \\
&= \langle \mathbf{x}]\vec{\mathbf{x}} \ ; \ (\vec{\boldsymbol{a}}^{-1} \uplus \vec{\boldsymbol{c}}^{-1}) \ ; \ ((\vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}) \oplus (\vec{\boldsymbol{c}} \ ; \ g \ ; \ \vec{\boldsymbol{d}}^{-1})) \ ; \ (\vec{\boldsymbol{b}} \uplus \vec{\boldsymbol{d}}) \ ; \ \vec{\mathbf{y}}\langle \mathbf{y}] \\
&= \langle \mathbf{x}][\mathbf{x}|\boldsymbol{a}+\boldsymbol{c}] \ ; \ ((\vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}) \oplus (\vec{\boldsymbol{c}} \ ; \ g \ ; \ \vec{\boldsymbol{d}}^{-1})) \ ; \ \langle \boldsymbol{b}+\boldsymbol{d}|\mathbf{y}\rangle\langle \mathbf{y}] \\
&= \langle \boldsymbol{a}+\boldsymbol{c}](\vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}}^{-1}) \oplus (\vec{\boldsymbol{c}} \ ; \ g \ ; \ \vec{\boldsymbol{d}}^{-1})[\boldsymbol{b}+\boldsymbol{d}\rangle \\
&= \langle \boldsymbol{a}+\boldsymbol{c}][\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!] \uplus [\![\boldsymbol{c}\rangle\!\rangle g \langle\!\langle \boldsymbol{d}]\!][\boldsymbol{b}+\boldsymbol{d}\rangle \\
&= \langle \boldsymbol{a}][\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!][\boldsymbol{b}\rangle \uplus \langle \boldsymbol{c}][\![\boldsymbol{c}\rangle\!\rangle g \langle\!\langle \boldsymbol{d}]\!][\boldsymbol{d}\rangle \\
&= H(f) \oplus H(g)
\end{aligned}
$$

For the identity and σ, it is trivial to see that H is a homomorphism.

Having shown H is a homomorphism, we now show that G and H are isomorphisms.

$$
G \circ H(f) = G(\langle \boldsymbol{a}][\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!][\boldsymbol{b}\rangle) = \langle\!\langle \boldsymbol{a}]\!][\![\boldsymbol{a}\rangle\!\rangle f \langle\!\langle \boldsymbol{b}]\!][\![\boldsymbol{b}\rangle\!\rangle = \vec{\boldsymbol{a}}^{-1} \ ; \ \vec{\boldsymbol{a}} \ ; \ f \ ; \ \vec{\boldsymbol{b}} \ ; \ \vec{\boldsymbol{b}}^{-1} = f
$$

$$
\begin{aligned}
H \circ G(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) &= H(\langle\!\langle \boldsymbol{a}]\!]f[\![\boldsymbol{b}\rangle\!\rangle) \\
&= \langle \mathbf{c}][\![\boldsymbol{c}\rangle\!\rangle\langle\!\langle \boldsymbol{a}]\!]f[\![\boldsymbol{b}\rangle\!\rangle\langle\!\langle \boldsymbol{d}]\!][\boldsymbol{d}\rangle \\
&= \langle \mathbf{c}][\mathbf{c}|\boldsymbol{a}] \ ; \ f \ ; \ [\boldsymbol{b}|\boldsymbol{d}][\boldsymbol{d}\rangle = \langle \boldsymbol{a}]f[\boldsymbol{b}\rangle
\end{aligned}
$$

$\square$

Remark 7.31. In a similar vein, we can show that injections $\mathbb{I}$, surjections $\mathbb{S}$, bijections $\mathbb{B}$, partial functions $\mathbb{P}$ and relations $\mathbb{R}$, are isomorphic to $\underset{\sim}{\mathrm{ORD}}(\mathrm{n}\mathbb{I})$, $\underset{\sim}{\mathrm{ORD}}(\mathrm{n}\mathbb{S})$, $\underset{\sim}{\mathrm{ORD}}(\mathrm{n}\mathbb{B})$, etc.

Lemma 7.32. *The following equations can be derived from the ones defined in prop. 7.26 and prop. 7.29:*

$$
[\boldsymbol{a}\rangle \, f \ ; \ \langle \boldsymbol{b}|\boldsymbol{b}'\rangle \ ; \ g \, \langle \boldsymbol{c}] = [\boldsymbol{a}\rangle \, f \, \langle \boldsymbol{b}] \ ; \ [\boldsymbol{b}'\rangle \, g \, \langle \boldsymbol{c}] \tag{41-1}
$$

$$
[\boldsymbol{a}\rangle \langle \boldsymbol{b}|\boldsymbol{b}'\rangle \langle \boldsymbol{c}] = [\boldsymbol{a}|\boldsymbol{b}] \ ; \ [\boldsymbol{b}'|\boldsymbol{c}] \tag{41-2}
$$

$$
\langle \boldsymbol{a}] \, f \ ; \ [\boldsymbol{b}|\boldsymbol{c}] \ ; \ g \, [\boldsymbol{d}\rangle = \langle \boldsymbol{a}] \, f \, [\boldsymbol{b}\rangle \ ; \ \langle \boldsymbol{c}] \, g \, [\boldsymbol{d}\rangle \tag{41-3}
$$

$$
\langle \boldsymbol{a}] \, [\boldsymbol{a}'|\boldsymbol{b}'] \, [\boldsymbol{b}\rangle = \langle \boldsymbol{a}|\boldsymbol{a}'\rangle \ ; \ \langle \boldsymbol{b}'|\boldsymbol{b}\rangle \tag{41-4}
$$

$$
\langle \boldsymbol{a}][\boldsymbol{a}\rangle \, f \, \langle \boldsymbol{b}][\boldsymbol{b}\rangle = \langle \mathbf{c}][\mathbf{c}\rangle \, f \, \langle \boldsymbol{d}][\boldsymbol{d}\rangle \tag{41-5}
$$

Proof.

$$\begin{aligned}
[\boldsymbol{a}\rangle f \;;\; \langle \boldsymbol{b}|\boldsymbol{b}'\rangle \;;\; g\langle \boldsymbol{c}] &= [\boldsymbol{a}\rangle f \;;\; \langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{b}'] \;;\; [\boldsymbol{b}'\rangle g\,\langle \boldsymbol{c}] && \text{(NOM-1)}\\
&= [\boldsymbol{a}\rangle f\,\langle \boldsymbol{b}] \;;\; [\boldsymbol{b}'|\boldsymbol{b}'] \;;\; [\boldsymbol{b}'\rangle g\,\langle \boldsymbol{c}] && \text{(NOM-5)}\\
&= [\boldsymbol{a}\rangle f\,\langle \boldsymbol{b}] \;;\; [\boldsymbol{b}'\rangle g\,\langle \boldsymbol{c}] && (\delta_{aa} = id_a)
\end{aligned}$$

$$\begin{aligned}
\langle \boldsymbol{a}]\, f \;;\; [\boldsymbol{b}|\boldsymbol{c}] \;;\; g\,[\boldsymbol{d}\rangle &= \langle \boldsymbol{a}]\, f \;;\; [\boldsymbol{b}|\boldsymbol{c}]\,[\boldsymbol{c}\rangle \;;\; \langle \boldsymbol{c}]\, g\,[\boldsymbol{d}\rangle && \text{(ORD-1)}\\
&= \langle \boldsymbol{a}]\, f\,[\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{c}|\boldsymbol{c}\rangle \;;\; \langle \boldsymbol{c}]\, g\,[\boldsymbol{d}\rangle && \text{(ORD-5)}\\
&= \langle \boldsymbol{a}]\, f\,[\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{c}]\, g\,[\boldsymbol{d}\rangle
\end{aligned}$$

The choice of $\boldsymbol{a}, \boldsymbol{b}$ is arbitrary, because we can prove that for any other choice $\boldsymbol{c}, \boldsymbol{d}$, we have $\langle \boldsymbol{a}][\boldsymbol{a}\rangle\, f\,\langle \boldsymbol{b}][\boldsymbol{b}\rangle = \langle \boldsymbol{c}][\boldsymbol{c}\rangle\, f\,\langle \boldsymbol{d}][\boldsymbol{d}\rangle$:

$$\begin{aligned}
\langle \boldsymbol{a}][\boldsymbol{a}\rangle\, f\,\langle \boldsymbol{b}][\boldsymbol{b}\rangle &= \langle \boldsymbol{a}](\,[\boldsymbol{a}\rangle\langle \boldsymbol{c}|\boldsymbol{c}\rangle \;;\; f \;;\; \langle \boldsymbol{d}|\boldsymbol{d}\rangle\,\langle \boldsymbol{b}]\,)[\boldsymbol{b}\rangle && (id \;;\; a = a = a \;;\; id)\\
&= \langle \boldsymbol{a}](\,[\boldsymbol{a}|\boldsymbol{c}] \;;\; [\boldsymbol{c}\rangle\, f \;;\; \langle \boldsymbol{d}|\boldsymbol{d}\rangle\,\langle \boldsymbol{b}]\,)[\boldsymbol{b}\rangle && \text{(NOM-4)}\\
&= \langle \boldsymbol{a}|\boldsymbol{a}\rangle \;;\; \langle \boldsymbol{c}]([\boldsymbol{c}\rangle\, f \;;\; \langle \boldsymbol{d}|\boldsymbol{d}\rangle\,\langle \boldsymbol{b}]\,)[\boldsymbol{b}\rangle && \text{(ORD-4)}\\
&= \langle \boldsymbol{c}](\,[\boldsymbol{c}\rangle\, f \;;\; \langle \boldsymbol{d}|\boldsymbol{d}\rangle\,\langle \boldsymbol{b}]\,)[\boldsymbol{b}\rangle && (id \;;\; a = a = a \;;\; id)\\
&= \langle \boldsymbol{c}](\,[\boldsymbol{c}\rangle\, f\,\langle \boldsymbol{d}] \;;\; [\boldsymbol{d}|\boldsymbol{b}]\,)[\boldsymbol{b}\rangle && \text{(NOM-5)}\\
&= \langle \boldsymbol{c}]\;\; [\boldsymbol{c}\rangle\, f\,\langle \boldsymbol{d}]\;\; [\boldsymbol{d}\rangle \;;\; \langle \boldsymbol{b}|\boldsymbol{b}\rangle && \text{(ORD-5)}\\
&= \langle \boldsymbol{c}][\boldsymbol{c}\rangle\, f\,\langle \boldsymbol{d}][\boldsymbol{d}\rangle && (id \;;\; a = a = a \;;\; id)
\end{aligned}$$

□

Remark 7.33. While technically $[\boldsymbol{a}\rangle\, f\,\langle \boldsymbol{b}]$ is just an operation symbol, the intended meaning is that $\langle \boldsymbol{a}]$ is mapping the index $\boldsymbol{i}$ to the name $\boldsymbol{a}_i$ and that $[\boldsymbol{a}\rangle$ is mapping the name $\boldsymbol{a}_i$ to the index $\boldsymbol{i}$.

Proposition 7.34. NOM : PROP → nPROP is a functor mapping an arrow of PROPs $F : \mathcal{S} \to \mathcal{S}$ to an arrow of nPROPs $\mathrm{NOM}(F) : \mathrm{NOM}(\mathcal{S}) \to \mathrm{NOM}(\mathcal{S})$ defined by

$$\mathrm{NOM}(F)([\boldsymbol{a}\rangle\, g\,\langle \boldsymbol{b}]) = [\boldsymbol{a}\rangle\, Fg\,\langle \boldsymbol{b}]. \tag{NOM-F}$$

Proof. We need to show that $\mathrm{NOM}(F)$ is equivariant and preserves bijections, sequential and parallel composition.

$$
\begin{aligned}
\pi \cdot \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f\langle \boldsymbol{b}]) &= \pi \cdot [\boldsymbol{a}\rangle Ff\langle \boldsymbol{b}] && (\text{NOM-F})\\
&= [\pi(\boldsymbol{a})\rangle Ff\langle \pi(\boldsymbol{b})] && (\pi\text{-def})\\
&= \underset{\sim}{\mathrm{NOM}}(F)([\pi(\boldsymbol{a})\rangle f\langle \pi(\boldsymbol{b})]) && (\text{NOM-F})\\
&= \underset{\sim}{\mathrm{NOM}}(F)(\pi \cdot [\boldsymbol{a}\rangle f\langle \boldsymbol{b}]) && (\pi\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
\underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{c}|\boldsymbol{c}']) &= \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{c}\rangle \mathrm{id}\langle \boldsymbol{c}']) && (\text{NOM-3})\\
&= [\boldsymbol{c}\rangle F\mathrm{id}\langle \boldsymbol{c}'] && (\text{NOM-F})\\
&= [\boldsymbol{c}\rangle \mathrm{id}\langle \boldsymbol{c}'] && (Fid = id)\\
&= [\boldsymbol{c}|\boldsymbol{c}'] && (\text{NOM-3})
\end{aligned}
$$

$$
\begin{aligned}
\underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f\langle \boldsymbol{c}] \,;\, [\boldsymbol{c}'\rangle g\langle \boldsymbol{b}]) &= \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f \,;\, \langle \boldsymbol{c}|\boldsymbol{c}'\rangle \,;\, g\langle \boldsymbol{b}]) && (41\text{-}1)\\
&= [\boldsymbol{a}\rangle F(f \,;\, \langle \boldsymbol{c}|\boldsymbol{c}'\rangle \,;\, g)\langle \boldsymbol{b}] && (\text{NOM-F})\\
&= [\boldsymbol{a}\rangle Ff \,;\, F\langle \boldsymbol{c}|\boldsymbol{c}'\rangle \,;\, Fg\langle \boldsymbol{b}] && (F(a \,;\, b) = Fa \,;\, Fb)\\
&= [\boldsymbol{a}\rangle Ff \,;\, \langle \boldsymbol{c}|\boldsymbol{c}'\rangle \,;\, Fg\langle \boldsymbol{b}] && (F\sigma = \sigma)\\
&= [\boldsymbol{a}\rangle Ff\langle \boldsymbol{c}] \,;\, [\boldsymbol{c}'\rangle Fg\langle \boldsymbol{b}] && (41\text{-}1)\\
&= \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f\langle \boldsymbol{c}]) \,;\, \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{c}'\rangle g\langle \boldsymbol{b}]) && (\text{NOM-F})
\end{aligned}
$$

$$
\begin{aligned}
\underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f\langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle g\langle \boldsymbol{d}]) &= \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a} + \boldsymbol{c}\rangle f \oplus g\langle \boldsymbol{b} + \boldsymbol{d}]) && (\text{NOM-2})\\
&= [\boldsymbol{a} + \boldsymbol{c}\rangle F(f \oplus g)\langle \boldsymbol{b} + \boldsymbol{d}] && (\text{NOM-F})\\
&= [\boldsymbol{a} + \boldsymbol{c}\rangle Ff \oplus Fg\langle \boldsymbol{b} + \boldsymbol{d}] && (F(a \oplus b) = Fa \oplus Fb)\\
&= [\boldsymbol{a}\rangle Ff\langle \boldsymbol{b}] \uplus [\boldsymbol{c}\rangle Fg\langle \boldsymbol{d}] && (\text{NOM-2})\\
&= \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{a}\rangle f\langle \boldsymbol{b}]) \uplus \underset{\sim}{\mathrm{NOM}}(F)([\boldsymbol{c}\rangle g\langle \boldsymbol{d}]) && (\text{NOM-F})
\end{aligned}
$$

□

Proposition 7.35. $\underset{\sim}{\mathrm{ORD}}$ is a functor mapping an arrow of $\underset{\sim}{\mathrm{nPROP}}$s $F : \mathcal{T} \to \mathcal{T}$ to an arrow of $\underset{\sim}{\mathrm{PROP}}$s $\underset{\sim}{\mathrm{ORD}}(F) : \underset{\sim}{\mathrm{ORD}}(\mathcal{T}) \to \underset{\sim}{\mathrm{ORD}}(\mathcal{T})$ defined by

$$\underset{\sim}{\mathrm{ORD}}(F)(\langle \mathbf{a}]\, f\, [\mathbf{b}\rangle) = \langle \mathbf{a}]\, Ff\, [\mathbf{b}\rangle \tag{ORD-F}$$

Proof. We need to show that ORD(F) preserves bijections, sequential and parallel composition.

$$\begin{aligned}
\mathrm{ORD}(F)(\langle \mathbf{c}|\mathbf{c}'\rangle) &= \mathrm{NOM}(F)(\langle \mathbf{c}]\mathrm{id}[\mathbf{c}'\rangle) && \text{(ORD-3)}\\
&= \langle \mathbf{c}]F\mathrm{id}[\mathbf{c}'\rangle && \text{(ORD-F)}\\
&= \langle \mathbf{c}]\mathrm{id}[\mathbf{c}'\rangle && (Fid = id)\\
&= \langle \mathbf{c}|\mathbf{c}'\rangle && \text{(ORD-3)}
\end{aligned}$$

$$\begin{aligned}
\mathrm{ORD}(F)(\langle \mathbf{a}]f[\mathbf{c}\rangle \,;\, \langle \mathbf{d}]g[\mathbf{b}\rangle) &= \mathrm{ORD}(F)(\langle \mathbf{a}]f \,;\, [\mathbf{c}|\mathbf{d}] \,;\, g[\mathbf{b}\rangle) && \text{(41-3)}\\
&= \langle \mathbf{a}]F(f \,;\, [\mathbf{c}|\mathbf{d}] \,;\, g)[\mathbf{b}\rangle && \text{(ORD-F)}\\
&= \langle \mathbf{a}]Ff \,;\, F[\mathbf{c}|\mathbf{d}] \,;\, Fg[\mathbf{b}\rangle && (F(a \,;\, b) = Fa \,;\, Fb)\\
&= \langle \mathbf{a}]Ff \,;\, [\mathbf{c}|\mathbf{d}] \,;\, Fg[\mathbf{b}\rangle && (F\delta = \delta)\\
&= \langle \mathbf{a}]Ff[\mathbf{c}\rangle \,;\, \langle \mathbf{d}]Fg[\mathbf{b}\rangle && \text{(41-3)}\\
&= \mathrm{ORD}(F)(\langle \mathbf{a}]f[\mathbf{c}\rangle) \,;\, \mathrm{ORD}(F)(\langle \mathbf{d}]g[\mathbf{b}\rangle) && \text{(ORD-F)}
\end{aligned}$$

$$\begin{aligned}
\mathrm{ORD}(F)(\langle \mathbf{a}]f[\mathbf{b}\rangle \oplus \langle \mathbf{c}]g[\mathbf{d}\rangle) &= \mathrm{ORD}(F)(\langle \mathbf{a}+\mathbf{c}]f \uplus g[\mathbf{b}+\mathbf{d}\rangle) && \text{(ORD-2)}\\
&= \langle \mathbf{a}+\mathbf{c}]F(f \uplus g)[\mathbf{b}+\mathbf{d}\rangle && \text{(ORD-F)}\\
&= \langle \mathbf{a}+\mathbf{c}]Ff \uplus Fg[\mathbf{b}+\mathbf{d}\rangle && (F(a \uplus b) = Fa \uplus Fb)\\
&= \langle \mathbf{a}]Ff[\mathbf{b}\rangle \oplus \langle \mathbf{c}]Fg[\mathbf{d}\rangle && \text{(ORD-2)}\\
&= \mathrm{ORD}(F)(\langle \mathbf{a}]f[\mathbf{b}\rangle) \oplus \mathrm{ORD}(F)(\langle \mathbf{c}]g[\mathbf{d}\rangle) && \text{(ORD-F)}
\end{aligned}$$

□

The next proposition has a variation in which we take PROPs in the weaker sense of Lack [60]. Then the unit $\mathcal{S} \to$ ORD(NOM($\mathcal{S}$)) is not an iso. To see where we need to be careful, the next example illustrates how the commutativity of $\uplus$ in an nPROP translates into the naturality of the symmetries in a PROP.

Example 7.36. [Commutativity of $\uplus$ translates to naturality of symmetries]

If $\mathcal{S}$ is a PROP in the sense of Lack [60] generated by a 'lollipop' $\lambda : 0 \to 1$ then we can show that $\lambda \oplus \mathrm{id}$ and $(\mathrm{id} \oplus \lambda) ; \sigma_{1,1}$ in $\mathcal{S}$ are sent to the same arrow in ORD(NOM($\mathcal{S}$)), namely we can show $\langle a][a\rangle\lambda \oplus \mathrm{id}\langle b, c][b, c\rangle = \langle a][a\rangle(\mathrm{id} \oplus \lambda) ; \sigma_{1,1}\langle b, c][b, c\rangle$:

$$
\begin{aligned}
\langle a][a\rangle\lambda \oplus \mathrm{id}\langle b, c][b, c\rangle &= \langle a][\rangle\lambda\langle b] \uplus [a\rangle\mathrm{id}\langle c][b, c\rangle && \text{(NOM-2)} \\
&= \langle a][a\rangle\mathrm{id}\langle c] \uplus [\rangle\lambda\langle b][b, c\rangle && \text{(NMT-comm)} \\
&= \langle a][a\rangle\mathrm{id} \oplus \lambda\langle c, b][b, c\rangle && \text{(NOM-2)} \\
&= \langle a][a\rangle\mathrm{id} \oplus \lambda\langle c, b] ; [b, c|b, c][b, c\rangle && (a = a ; id, \delta_{aa} = id_a) \\
&= \langle a][a\rangle(\mathrm{id} \oplus \lambda) ; \langle c, b|b, c\rangle\langle b, c][b, c\rangle && \text{(NOM-5)} \\
&= \langle a][a\rangle(\mathrm{id} \oplus \lambda) ; \sigma_{1,1}\langle b, c][b, c\rangle && (\sigma\text{-def})
\end{aligned}
$$

which is an instance of (SMT-nat) and does not hold in $\mathcal{S}$.

As we can see from the example, the naturality of symmetries in a PROP is necessary in order to obtain that $\mathcal{S} \to$ ORD(NOM($\mathcal{S}$)) is an iso in the next proposition.

Proposition 7.37. For each PROP $\mathcal{S}$, there is an isomorphism of PROPs, natural in $\mathcal{S}$,

$$\Delta : \mathcal{S} \to \mathrm{ORD}(\mathrm{NOM}(\mathcal{S}))$$

mapping $f \in \mathcal{S}$ to $\langle \boldsymbol{a}][\boldsymbol{a}\rangle\, f\, \langle \boldsymbol{b}][\boldsymbol{b}\rangle$ for some choice of $\boldsymbol{a}, \boldsymbol{b}$.

Proof. We first show that Δ is a homomorphism, i.e. it preserves symmetries and the two compositions. Δ must also preserve the equations of a PROP.

$$
\begin{aligned}
\Delta(\langle \mathbf{x}|\mathbf{x}'\rangle) &= \langle \boldsymbol{a}]\,[\boldsymbol{a}\rangle\,\langle \mathbf{x}|\mathbf{x}'\rangle\,\langle \boldsymbol{b}]\,[\boldsymbol{b}\rangle && (\Delta\text{-def}) \\
&= \langle \boldsymbol{a}](\,[\boldsymbol{a}|\mathbf{x}]\ ;\ [\mathbf{x}'|\boldsymbol{b}]\,)[\boldsymbol{b}\rangle && \text{(41-2)} \\
&= \langle \boldsymbol{a}|\boldsymbol{a}\rangle\ ;\ \langle \mathbf{x}]\ \ [\mathbf{x}'|\boldsymbol{b}]\ \ [\boldsymbol{b}\rangle && \text{(ORD-4)} \\
&= \langle \mathbf{x}]\,[\mathbf{x}'|\boldsymbol{b}]\,[\boldsymbol{b}\rangle && (id\ ;\ a = a = a\ ;\ id) \\
&= \langle \mathbf{x}|\mathbf{x}'\rangle\ ;\ \langle \boldsymbol{b}|\boldsymbol{b}\rangle && \text{(ORD-5)} \\
&= \langle \mathbf{x}|\mathbf{x}'\rangle && (id\ ;\ a = a = a\ ;\ id)
\end{aligned}
$$

$$
\begin{aligned}
\Delta(f \oplus g) &= \langle \boldsymbol{a}_f + \boldsymbol{a}_g] [\boldsymbol{a}_f + \boldsymbol{a}_g \rangle\, f \oplus g \,\langle \boldsymbol{b}_f + \boldsymbol{b}_g]\, [\boldsymbol{b}_f + \boldsymbol{b}_g \rangle && (\Delta\text{-def, 41-5}) \\
&= \langle \boldsymbol{a}_f + \boldsymbol{a}_g]([\boldsymbol{a}_f \rangle\, f \,\langle \boldsymbol{b}_f] \uplus [\boldsymbol{a}_g \rangle\, g \,\langle \boldsymbol{b}_g])[\boldsymbol{b}_f + \boldsymbol{b}_g \rangle && (\text{NOM-2}) \\
&= \langle \boldsymbol{a}_f][\boldsymbol{a}_f \rangle\, f \,\langle \boldsymbol{b}_f][\boldsymbol{b}_f \rangle \oplus \langle \boldsymbol{a}_g][\boldsymbol{a}_g \rangle\, g \,\langle \boldsymbol{b}_g][\boldsymbol{b}_g \rangle && (\text{ORD-2}) \\
&= \Delta(f) \oplus \Delta(g) && (\Delta\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
\Delta(f \,;\, g) &= \langle \boldsymbol{a}]([\boldsymbol{a} \rangle f \,;\, g \langle \boldsymbol{b}])[\boldsymbol{b} \rangle && (\Delta\text{-def}) \\
&= \langle \boldsymbol{a}]([\boldsymbol{a} \rangle f \langle \boldsymbol{c}] \,;\, [\boldsymbol{c} \rangle g \langle \boldsymbol{b}])[\boldsymbol{b} \rangle && (\text{NOM-1}) \\
&= \langle \boldsymbol{a}][\boldsymbol{a} \rangle\, f \,\langle \boldsymbol{c}][\boldsymbol{c} \rangle \,;\, \langle \boldsymbol{c}][\boldsymbol{c} \rangle\, g \,\langle \boldsymbol{b}][\boldsymbol{b} \rangle && (\text{ORD-1}) \\
&= \Delta(f) \,;\, \Delta(g) && (\Delta\text{-def})
\end{aligned}
$$

In the last step of the derivation above, we used the fact that we can arbitrarily choose $\boldsymbol{a}, \boldsymbol{b}$ in Δ, which follows form the last equation of lem. 7.32.

To show that there is an isomorphism between $\mathcal{S}$ and ORD(NOM($\mathcal{S}$)), we define an inverse to Δ:

$$
\Gamma : \text{ORD}(\text{NOM}(\mathcal{S})) \to \mathcal{S}
$$

mapping the $\langle \boldsymbol{a}'][\boldsymbol{a} \rangle\, f \,\langle \boldsymbol{b}][\boldsymbol{b}' \rangle \in$ ORD(NOM($\mathcal{S}$)) generated by an $f \in \mathcal{S}$ to $\langle \boldsymbol{a}' | \boldsymbol{a} \rangle \,;\, f \,;\, \langle \boldsymbol{b} | \boldsymbol{b}' \rangle$, such that that $\Delta \circ \Gamma$ and $\Gamma \circ \Delta$ are identities.

However, in order for Γ to be well-defined, we also need to show that it is a homomorphism. Since a homomorphism between PROPs needs to preserve equations, the equation (SMT-nat) in ORD(NOM($\mathcal{S}$)) must be derivable in $\mathcal{S}$. This is obviously impossible for $\mathcal{S}$ a la Lack (see the example above). In the converse case we have:

$$
\begin{aligned}
\Gamma(\langle \boldsymbol{a} | \boldsymbol{a}' \rangle) &= \Gamma(\langle \boldsymbol{a}][\boldsymbol{a}' | \boldsymbol{a}'][\boldsymbol{a}' \rangle) && (id \,;\, a = a = a \,;\, id,\ \text{41-4}) \\
&= \Gamma(\langle \boldsymbol{a}][\boldsymbol{a}' \rangle id \langle \boldsymbol{a}'][\boldsymbol{a}' \rangle) && (\text{NOM-3}) \\
&= \langle \boldsymbol{a} | \boldsymbol{a}' \rangle \,;\, id \,;\, \langle \boldsymbol{a}' | \boldsymbol{a}' \rangle && (\Gamma\text{-def}) \\
&= \langle \boldsymbol{a} | \boldsymbol{a}' \rangle && (id \,;\, a = a = a \,;\, id)
\end{aligned}
$$

$$
\begin{aligned}
&\Gamma(\langle \boldsymbol{a}_f][\boldsymbol{a}'_f\rangle\, f\, \langle \boldsymbol{b}'_f][\boldsymbol{b}_f\rangle \oplus \langle \boldsymbol{a}_g][\boldsymbol{a}'_g\rangle\, g\, \langle \boldsymbol{b}'_g][\boldsymbol{b}_g\rangle) \\
&= \Gamma(\langle \boldsymbol{a}_f + \boldsymbol{a}_g](\, [\boldsymbol{a}'_f\rangle\, f\, \langle \boldsymbol{b}'_f] \;\uplus\; [\boldsymbol{a}'_g\rangle\, g\, \langle \boldsymbol{b}'_g]\,)[\boldsymbol{b}_f + \boldsymbol{b}_g\rangle) && (\text{ORD-2}) \\
&= \Gamma(\langle \boldsymbol{a}_f + \boldsymbol{a}_g](\, [\boldsymbol{a}'_f + \boldsymbol{a}'_g\rangle\, f \oplus g\, \langle \boldsymbol{b}'_f + \boldsymbol{b}'_g]\,)[\boldsymbol{b}_f + \boldsymbol{b}_g\rangle) && (\text{NOM-2}) \\
&= \langle \boldsymbol{a}_f + \boldsymbol{a}_g | \boldsymbol{a}'_f + \boldsymbol{a}'_g\rangle \;;\; (f \oplus g) \;;\; \langle \boldsymbol{b}'_f + \boldsymbol{b}'_g | \boldsymbol{b}_f + \boldsymbol{b}_g\rangle && (\Gamma\text{-def}) \\
&= (\langle \boldsymbol{a}_f | \boldsymbol{a}'_f\rangle \oplus \langle \boldsymbol{a}_g | \boldsymbol{a}'_g\rangle) \;;\; (f \oplus g) \;;\; (\langle \boldsymbol{b}'_f | \boldsymbol{b}_f\rangle \oplus \langle \boldsymbol{b}'_g | \boldsymbol{b}_g\rangle) \\
&= (\langle \boldsymbol{a}_f | \boldsymbol{a}'_f\rangle \;;\; f \;;\; \langle \boldsymbol{b}'_f | \boldsymbol{b}_f\rangle) \oplus (\langle \boldsymbol{a}_g | \boldsymbol{a}'_g\rangle \;;\; g \;;\; \langle \boldsymbol{b}'_g | \boldsymbol{b}_g\rangle) && (\text{SMT-ch}) \\
&= \Gamma(\langle \boldsymbol{a}_f][\boldsymbol{a}'_f\rangle\, f\, \langle \boldsymbol{b}'_f][\boldsymbol{b}_f\rangle) \;\oplus\; \Gamma(\langle \boldsymbol{a}_g][\boldsymbol{a}'_g\rangle\, g\, \langle \boldsymbol{b}'_g][\boldsymbol{b}_g\rangle) && (\Gamma\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
\Gamma(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'][\boldsymbol{b}\rangle \,;\, \langle \boldsymbol{c}][\boldsymbol{c}'\rangle\, g\, \langle \boldsymbol{d}'][\boldsymbol{d}\rangle) &= \Gamma(\langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'] \;;\; [\boldsymbol{b} | \boldsymbol{c}] \;;\; [\boldsymbol{c}'\rangle\, g\, \langle \boldsymbol{d}']\,)[\boldsymbol{d}'\rangle) && (41\text{-}3) \\
&= \Gamma(\langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle (\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) \langle \boldsymbol{c}] \;;\; [\boldsymbol{c}'\rangle\, g\, \langle \boldsymbol{d}']\,)[\boldsymbol{d}'\rangle) && (\text{NOM-5}) \\
&= \Gamma(\langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle\, f \;;\; \langle \boldsymbol{b}' | \boldsymbol{b}\rangle \;;\; \langle \boldsymbol{c} | \boldsymbol{c}'\rangle \;;\; g\, \langle \boldsymbol{d}']\,)[\boldsymbol{d}'\rangle) && (41\text{-}1) \\
&= \langle \boldsymbol{a} | \boldsymbol{a}'\rangle \;;\; f \;;\; \langle \boldsymbol{b}' | \boldsymbol{b}\rangle \;;\; \langle \boldsymbol{c} | \boldsymbol{c}'\rangle \;;\; g \;;\; \langle \boldsymbol{d}' | \boldsymbol{d}'\rangle && (\Gamma\text{-def}) \\
&= \Gamma(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'][\boldsymbol{b}\rangle) \,;\, \Gamma(\langle \boldsymbol{c}][\boldsymbol{c}'\rangle\, g\, \langle \boldsymbol{d}'][\boldsymbol{d}\rangle) && (\Gamma\text{-def})
\end{aligned}
$$

Finally, we verify that $\Delta \circ \Gamma = Id_{\mathrm{ORD}(\mathrm{NOM}(\mathcal{S}))}$ and $\Gamma \circ \Delta = Id_{\mathcal{S}}$:

$$
\begin{aligned}
\Delta(\Gamma(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'][\boldsymbol{b}\rangle)) &= \Delta(\, \langle \boldsymbol{a} | \boldsymbol{a}'\rangle \,;\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) && (\Gamma\text{-def}) \\
&= \langle \boldsymbol{x}]\, (\, [\boldsymbol{x}\rangle (\, \langle \boldsymbol{a} | \boldsymbol{a}'\rangle \,;\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) \langle \boldsymbol{y}]\,)\, [\boldsymbol{y}\rangle && (\Delta\text{-def}) \\
&= \langle \boldsymbol{x}]\, (\, [\boldsymbol{x} | \boldsymbol{a}] \;;\; [\boldsymbol{a}'\rangle (\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) \langle \boldsymbol{y}]\,)\, [\boldsymbol{y}\rangle && (\text{NOM-4}) \\
&= \langle \boldsymbol{x} | \boldsymbol{x}\rangle \;;\; \langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle (\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) \langle \boldsymbol{y}]\,)\, [\boldsymbol{y}\rangle && (\text{ORD-4}) \\
&= \langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle (\, f \,;\, \langle \boldsymbol{b}' | \boldsymbol{b}\rangle\,) \langle \boldsymbol{y}]\,)[\boldsymbol{y}\rangle && (id \;;\; a = a = a \;;\; id) \\
&= \langle \boldsymbol{a}](\, [\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'] \;;\; [\boldsymbol{b} | \boldsymbol{y}]\,)[\boldsymbol{y}\rangle && (\text{NOM-5}) \\
&= \langle \boldsymbol{a}]\; [\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}']\; [\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{y} | \boldsymbol{y}\rangle && (\text{ORD-5}) \\
&= \langle \boldsymbol{a}][\boldsymbol{a}'\rangle\, f\, \langle \boldsymbol{b}'][\boldsymbol{b}\rangle && (id \;;\; a = a = a \;;\; id)
\end{aligned}
$$

$$
\begin{aligned}
\Gamma(\Delta(f)) &= \Gamma(\langle \boldsymbol{a}](\, [\boldsymbol{a}\rangle f \langle \boldsymbol{b}]\,)[\boldsymbol{b}\rangle) && (\Delta\text{-def}) \\
&= \langle \boldsymbol{a} | \boldsymbol{a}\rangle \;;\; f \;;\; \langle \boldsymbol{b} | \boldsymbol{b}\rangle && (\Gamma\text{-def}) \\
&= f && (id \;;\; a = a = a \;;\; id)
\end{aligned}
$$

□

Proposition 7.38. For each nPROP $\mathcal{T}$, there is an isomorphism of nPROPs, natural in $\mathcal{T}$,

$$\mathrm{NOM}(\mathrm{ORD}(\mathcal{T})) \to \mathcal{T}$$

mapping the $[\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{d}]$ generated by an $f : \underline{\boldsymbol{a}} \to \underline{\boldsymbol{b}}$ in $\mathcal{T}$ to $[\boldsymbol{c}|\boldsymbol{a}]\ ;\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]$.

Proof. We define a converse $\Delta_n : \mathcal{T} \to \mathrm{NOM}(\mathrm{ORD}(\mathcal{T}))$ mapping $f : \underline{\boldsymbol{a}} \to \underline{\boldsymbol{b}}$ to $[\boldsymbol{a}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{b}]$ for some choice of $\boldsymbol{a}, \boldsymbol{b}$.

We now verify that $\Gamma_n(\Delta_n(f)) = f$ for any f:

$$\begin{aligned}
\Gamma_n(\Delta_n(f)) &= \Gamma_n([\boldsymbol{a}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{b}]) && (\Delta_n\text{-def})\\
&= [\boldsymbol{a}|\boldsymbol{a}]\ ;\ f\ ;\ [\boldsymbol{b}|\boldsymbol{b}] && (\Gamma_n\text{-def})\\
&= f && (id\ ;\ a = a = a\ ;\ id)
\end{aligned}$$

and

$$\begin{aligned}
\Delta_n(\Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{d}])) &= \Delta_n(\,[\boldsymbol{c}|\boldsymbol{a}]\ ;\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\,) && (\Gamma_n\text{-def})\\
&= [\boldsymbol{c}\rangle(\ \langle\boldsymbol{c}](\ [\boldsymbol{c}|\boldsymbol{a}]\ ;\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\ )[\boldsymbol{d}\rangle\ )\langle\boldsymbol{d}] && (\Delta_n\text{-def})\\
&= [\boldsymbol{c}\rangle(\ \langle\boldsymbol{c}|\boldsymbol{c}\rangle\ ;\ \langle\boldsymbol{a}](\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\ )[\boldsymbol{d}\rangle\ )\langle\boldsymbol{d}] && (\text{ORD-4})\\
&= [\boldsymbol{c}\rangle(\ \langle\boldsymbol{a}](\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\ )[\boldsymbol{d}\rangle\ )\langle\boldsymbol{d}] && (id\ ;\ a = a = a\ ;\ id)\\
&= [\boldsymbol{c}\rangle(\ \langle\boldsymbol{a}]\ f\ [\boldsymbol{b}\rangle\ \ ;\ \ \langle\boldsymbol{d}|\boldsymbol{d}\rangle\ )\langle\boldsymbol{d}] && (\text{ORD-5})\\
&= [\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{d}] && (id\ ;\ a = a = a\ ;\ id)
\end{aligned}$$

We also show that Δ_n and Γ_n preserve the two kinds of composition and symmetries:

$$\begin{aligned}
\Delta_n(f\ ;\ g) &= [\boldsymbol{a}\rangle(\ \langle\boldsymbol{a}]f\ ;\ g[\boldsymbol{b}\rangle\ )\langle\boldsymbol{b}] && (\Delta_n\text{-def})\\
&= [\boldsymbol{a}\rangle(\ \langle\boldsymbol{a}]f[\boldsymbol{c}\rangle\ ;\ \langle\boldsymbol{c}]g[\boldsymbol{b}\rangle\ )\langle\boldsymbol{b}] && (\text{ORD-1})\\
&= [\boldsymbol{a}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{c}\rangle\langle\boldsymbol{c}]\ \ ;\ \ [\boldsymbol{c}\rangle\langle\boldsymbol{c}]\, g\, [\boldsymbol{b}\rangle\langle\boldsymbol{b}] && (\text{NOM-1})\\
&= \Delta_n(f)\ ;\ \Delta_n(g) && (\Delta_n\text{-def})
\end{aligned}$$

$$
\begin{aligned}
\Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{d}]\ ;\ [\boldsymbol{d}'\rangle\langle\boldsymbol{e}]\, g\, [\boldsymbol{f}\rangle\langle\boldsymbol{h}]) &= \Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\ ;\ \langle\boldsymbol{d}|\boldsymbol{d}'\rangle\ ;\ \langle\boldsymbol{e}]\, g\, [\boldsymbol{f}\rangle\langle\boldsymbol{h}]) && \text{(ORD-1)}\\
&= \Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, (f\ ;\ [\boldsymbol{b}|\boldsymbol{d}])\, [\boldsymbol{d}'\rangle\ ;\ \langle\boldsymbol{e}]\, g\, [\boldsymbol{f}\rangle\langle\boldsymbol{h}]) && \text{(ORD-5)}\\
&= \Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, (f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\ ;\ [\boldsymbol{d}'|\boldsymbol{e}]\ ;\ g)\, [\boldsymbol{f}\rangle\langle\boldsymbol{h}]) && \text{(41-3)}\\
&= [\boldsymbol{c}|\boldsymbol{a}]\ ;\ f\ ;\ [\boldsymbol{b}|\boldsymbol{d}]\ ;\ [\boldsymbol{d}'|\boldsymbol{e}]\ ;\ g\ ;\ [\boldsymbol{f}|\boldsymbol{h}] && (\Gamma_n\text{-def})\\
&= \Gamma_n([\boldsymbol{c}\rangle\langle\boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle\langle\boldsymbol{d}])\ ;\ \Gamma_n([\boldsymbol{d}'\rangle\langle\boldsymbol{e}]\, g\, [\boldsymbol{f}\rangle\langle\boldsymbol{h}]) && (\Gamma_n\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
\Delta_n(f \uplus g) &= [\boldsymbol{a}_f + \boldsymbol{a}_g\rangle\, \langle\boldsymbol{a}_f + \boldsymbol{a}_g]\, f \uplus g\, [\boldsymbol{b}_f + \boldsymbol{b}_g\rangle\, \langle\boldsymbol{b}_f + \boldsymbol{b}_g] && (\Delta_n\text{-def})\\
&= [\boldsymbol{a}_f + \boldsymbol{a}_g\rangle(\, \langle\boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle\ \oplus\ \langle\boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle\,)\langle\boldsymbol{b}_f + \boldsymbol{b}_g] && \text{(ORD-2)}\\
&= [\boldsymbol{a}_f\rangle\langle\boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle\langle\boldsymbol{b}_f]\ \uplus\ [\boldsymbol{a}_g\rangle\langle\boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle\langle\boldsymbol{b}_g] && \text{(NOM-2)}\\
&= \Delta_n(f) \uplus \Delta_n(g) && (\Delta_n\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
&\Gamma_n([\boldsymbol{a}'_f\rangle\langle\boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle\langle\boldsymbol{b}'_f] \uplus [\boldsymbol{a}'_g\rangle\langle\boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle\langle\boldsymbol{b}'_g])\\
&= \Gamma_n([\boldsymbol{a}'_f + \boldsymbol{a}'_g\rangle(\, \langle\boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle\ \oplus\ \langle\boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle\,)\langle\boldsymbol{b}'_f + \boldsymbol{b}'_g]) && \text{(NOM-2)}\\
&= \Gamma_n([\boldsymbol{a}'_f + \boldsymbol{a}'_g\rangle(\, \langle\boldsymbol{a}_f + \boldsymbol{a}_g]\, f \uplus g\, [\boldsymbol{b}_f + \boldsymbol{b}_g\rangle\,)\langle\boldsymbol{b}'_f + \boldsymbol{b}'_g]) && \text{(ORD-2)}\\
&= [\boldsymbol{a}'_f + \boldsymbol{a}'_g|\boldsymbol{a}_f + \boldsymbol{a}_g]\ ;\ (f \uplus g)\ ;\ [\boldsymbol{b}_f + \boldsymbol{b}_g|\boldsymbol{b}'_f + \boldsymbol{b}'_g] && (\Gamma_n\text{-def})\\
&= ([\boldsymbol{a}'_f|\boldsymbol{a}_f] \uplus [\boldsymbol{a}'_g|\boldsymbol{a}_g])\ ;\ (f \uplus g)\ ;\ ([\boldsymbol{b}_f|\boldsymbol{b}'_f] \uplus [\boldsymbol{b}_g|\boldsymbol{b}'_g])\\
&= ([\boldsymbol{a}'_f|\boldsymbol{a}_f]\ ;\ f\ ;\ [\boldsymbol{b}_f|\boldsymbol{b}'_f]) \uplus ([\boldsymbol{a}'_g|\boldsymbol{a}_g]\ ;\ g\ ;\ [\boldsymbol{b}_g|\boldsymbol{b}'_g]) && \text{(NMT-ch)}\\
&= \Gamma_n([\boldsymbol{a}'_f\rangle\langle\boldsymbol{a}_f]\, f\, [\boldsymbol{b}_f\rangle\langle\boldsymbol{b}'_f])\ \uplus\ \Gamma_n([\boldsymbol{a}'_g\rangle\langle\boldsymbol{a}_g]\, g\, [\boldsymbol{b}_g\rangle\langle\boldsymbol{b}'_g]) && (\Gamma_n\text{-def})
\end{aligned}
$$

$$
\begin{aligned}
\Delta_n([\boldsymbol{x}|\boldsymbol{y}]) &= [\boldsymbol{x}\rangle\langle\boldsymbol{x}]\, [\boldsymbol{x}|\boldsymbol{y}]\, [\boldsymbol{y}\rangle\langle\boldsymbol{y}] && (\Delta_n\text{-def})\\
&= [\boldsymbol{x}\rangle(\, \langle\boldsymbol{x}|\boldsymbol{x}\rangle\ ;\ \langle\boldsymbol{y}|\boldsymbol{y}\rangle\,)\langle\boldsymbol{y}] && \text{(41-4)}\\
&= [\boldsymbol{x}\rangle\ \mathrm{id}\ \langle\boldsymbol{y}]\\
&= [\boldsymbol{x}|\boldsymbol{y}] && \text{(NOM-3)}
\end{aligned}
$$

$$
\begin{aligned}
\Gamma_n([\boldsymbol{a}|\boldsymbol{b}]) &= \Gamma_n([\boldsymbol{a}\rangle id\langle\boldsymbol{b}]) && \text{(NOM-3)}\\
&= \Gamma_n([\boldsymbol{a}\rangle\langle\boldsymbol{b}]id[\boldsymbol{b}\rangle\langle\boldsymbol{b}]) && \text{(ORD-3)}\\
&= [\boldsymbol{a}|\boldsymbol{b}]\ ;\ id\ ;\ [\boldsymbol{b}|\boldsymbol{b}] && (\Gamma_n\text{-def})\\
&= [\boldsymbol{a}|\boldsymbol{b}]
\end{aligned}
$$

□

Since the last two propositions provide an isomorphic unit and counit of an adjunction, we obtain

Theorem 7.39. The categories PROP and nPROP are equivalent.

Remark 7.40. If we generalise the notion of PROP from MacLane [47] to Lack [60], in other words, if we drop the last equation of fig. 7.2 expressing the naturality of symmetries, we still obtain an adjunction, in which NOM is left-adjoint to ORD. Nominal PROPs then are a full reflective subcategory of ordinary PROPs. In other words, the (generalised) PROPs $\mathcal{S}$ that satisfy naturality of symmetries are exactly those for which $\mathcal{S} \cong \mathrm{ORD}(\mathrm{NOM}(\mathcal{S}))$.

7.6 Equivalence of theories

We should be able to switch easily between a notion of ordered names on the one hand and a notion of unordered abstract names on the other. This intuition is reinforced by putting fig. 7.3 and fig. 7.4 next to each other. A careful investigation suggests that there is a general procedure to automatically translate one into the other.

This section will give such translations and prove that these translations are inverse to each other and preserve completeness. This yields a tool to derive completeness of an NMT from the completeness of the corresponding SMT and vice versa.

We start with giving a more precise definition of the relation between an SMT/NMT and a PROP/nPROP.

We previously defined a theory of (nominal) string diagrams as the pair $\langle \Sigma, E\rangle$, where Σ is the set of generators and $E \subseteq \mathrm{Trm}(\Sigma) \times \mathrm{Trm}(\Sigma)$ is the set of equations. The operation $\mathrm{Prop} : \mathrm{SMT} \to \mathrm{PROP}$ takes the signature $\langle \Sigma, E\rangle$ to the category of SMT terms, quotiented by the equations of E together with the equations of an SMT.

Definition 7.41. The operation Prop : SMT → PROP is defined as

$$\mathrm{Prop}\,\langle\Sigma, E\rangle = \mathrm{Trm}(\Sigma)/\mathcal{Th}(E \cup \mathbf{SMT})$$

$$\frac{}{s = s \in \mathcal{Th}(E)} \qquad \frac{s = t \in \mathcal{Th}(E)}{t = s \in \mathcal{Th}(E)}$$

$$\frac{s = t \in \mathcal{Th}(E) \qquad t = u \in \mathcal{Th}(E)}{s = u \in \mathcal{Th}(E)} \qquad \frac{s = s' \in \mathcal{Th}(E) \qquad t = t' \in \mathcal{Th}(E)}{s * t = s' * t' \in \mathcal{Th}(E)}$$

Figure 7.9: Closure operator

This definition uses the closure operator $\mathcal{Th}$, defined in fig. 7.9. $\mathcal{Th}$ is the usual closure operator for equational deduction. We have $* = \{\ ;\ ,\ \oplus\}$ for $\mathcal{Th}$ over equations on Trms and for equations over nTrms we have $* = \{\ ;\ ,\ \uplus\}$ along with an additional rule for permutations:

$$\frac{s = t \in \mathcal{Th}(E)}{\pi \cdot s = \pi \cdot t \in \mathcal{Th}(E)}$$

We have a similar construction for NMTs, where we define a functor nProp : NMT → nPROP:

Definition 7.42. nProp : NMT → nPROP is defined as

$$\mathrm{nProp}\,\langle\Sigma, E\rangle = \mathrm{nTrm}(\Sigma)/\mathcal{Th}(E \cup \mathbf{NMT})$$

Finally, we prove the following property of the closure operator, which we will use in a later lemma.

Lemma 7.43. *Given a set of equations $X \subseteq \mathrm{nTrm}(A) \times \mathrm{nTrm}(A)$ (or $X \subseteq \mathrm{Trm}(A) \times \mathrm{Trm}(A)$), and a homomorphism $f : \mathrm{nTrm}(A) \to \mathrm{nTrm}(B)$ (or $f : \mathrm{Trm}(A) \to \mathrm{Trm}(B)$), we have:*

$$f[\mathcal{Th}(X)] \subseteq \mathcal{Th}(f[X])$$

Proof. The statement above is equivalent to $\forall (s,t) \in \mathcal{Th}(X)\,.\,(f(s), f(t)) \in \mathcal{Th}(f[X])$. Then, by induction on the formation rules of the set $\mathcal{Th}(X)$, we have the following cases:

- If $(s,t) \in X$, then $(f(s), f(t)) \in f[X]$, by definition and therefore, $(f(s), f(t)) \in \mathcal{Th}(f[X])$.

- If $(s,t) \in \mathcal{Th}(X)$, by reflexivity, symmetry or transitivity, then by IH $(f(s), f(t)) \in \mathcal{Th}(f[X])$.
- If $(s,t) \in \mathcal{Th}(X)$, by congruence of $;$ or $\uplus$ or permutation, the result follows by IH and the fact that f is a homomorphism, e.g.:

  For $(s \uplus t, s' \uplus t') \in \mathcal{Th}(X)$, by IH, we have

  $$(f(s), f(s')) \in \mathcal{Th}(f[X]) \text{ and } (f(t), f(t')) \in \mathcal{Th}(f[X])$$

  then we also have

  $$(f(s) \uplus f(t), f(s') \uplus f(t')) \in \mathcal{Th}(f[X])$$

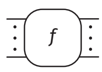
  and since we know f is a homomorphism, we have $f(s \uplus t) = f(s) \uplus f(t)$, thus

  $$(f(s \uplus t), f(s' \uplus t')) \in \mathcal{Th}(f[X])$$

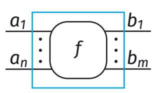
$\square$

7.6.1 Embedding PROPSs into nPROPs

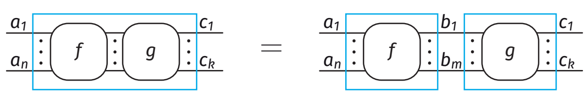
This section briefly returns to sec. 7.5, summarising equivalence of the categories PROP and nPROP by embedding ordinary PROPs into nPROPs. Recall that this is achieved in the following manner. Given an ordinary diagram $f : n \to m$,

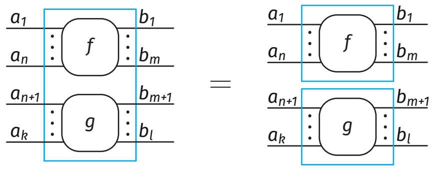


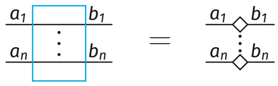
we create "boxed" nominal versions $[\boldsymbol{a}\rangle f\langle\boldsymbol{b}]$, where $\boldsymbol{a} = [a_1, \dots, a_n]$ and $\boldsymbol{b} = [b_1, \dots, b_m]$ are lists of pairwise distinct names:



For "boxing" to preserve the relevant structure, we have to ensure, in particular, that the symmetric monoidal tensor of a PROPs is mapped to the commutative tensor of nPROPs, and that sequential composition and identities are preserved:







We can thus build the following embedding of an SMT into an nPROP:

Given an SMT $\langle\Sigma, E\rangle$, we can generate an nPROP, by taking all the SMT-terms over Σ, as generators (taking $\mathsf{Trm}(\Sigma)$ to $n\mathsf{Trm}(\mathsf{Trm}(\Sigma))$) and taking $\mathsf{box}(E) \cup \mathbf{NOM}$ as equations, where:

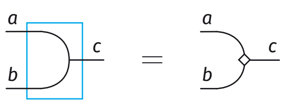
- $\mathsf{box}(E) = \{[\boldsymbol{a}\rangle f\langle \boldsymbol{b}] = [\boldsymbol{a}\rangle g\langle \boldsymbol{b}] \mid f = g \in E\}$
- **NOM** are the equations from prop. 7.26

7.6.2 Translating SMTs into NMTs

Whilst the construction above gives us a way of embedding equivalence classes of ordinary string diagrams into equivalence classes of nominal ones, it does not answer the question of how to translate the axioms defining an SMT into the axioms of the corresponding NMT.

If we recall the definition of an NMT, we see that the signature of a nominal theory consists of a set of ordinary generators Σ and set of equations over $n\mathsf{Trm}(\Sigma)$. Thus, given the ordinary signature of an SMT, with generators Σ and the set of equations $E \subseteq \mathsf{Trm}(\Sigma) \times \mathsf{Trm}(\Sigma)$, we need to obtain an $E' \subseteq n\mathsf{Trm}(\Sigma) \times n\mathsf{Trm}(\Sigma)$ such that any equivalence class induced by E' and the equations of NOM (due to the ordinary diagrams being embedded in nominal ones) are mirrored by E.

Intuitively, we translate equations of E, by first embedding them inside a "nominal box" as a whole and then use the rules of NOM to recursively normalise all *sub-diagrams* into nominal ones (see ex. 7.45). When we hit the base case, i.e. a "boxed" generator from Σ, we simply replace it with a corresponding nominal generator:



We perform this normalisation via the function $\mathsf{nfNmt} : n\mathsf{Trm}(\mathsf{Trm}(\Sigma)) \to n\mathsf{Trm}(\Sigma)$. In the definition below, we use the notation $\underline{\gamma}$ to highlight the difference between an element γ of Σ and the string diagram $\underline{\gamma} \in \mathsf{Trm}(\Sigma)$ as in the blue box above.

$$\mathrm{nfNmt}([\boldsymbol{a}\rangle\underline{\gamma}\langle\boldsymbol{b}]) = [\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}] \text{ where } \gamma \in \Sigma$$

$$\mathrm{nfNmt}([a\rangle id\langle b]) = \delta_{ab}$$

$$\mathrm{nfNmt}([ab\rangle\sigma\langle cd]) = [ab|dc]$$

$$\mathrm{nfNmt}([\boldsymbol{a}\rangle f \ ; \ g\langle\boldsymbol{c}]) = \mathrm{nfNmt}([\boldsymbol{a}\rangle f\langle\boldsymbol{b}]) \ ; \ \mathrm{nfNmt}([\boldsymbol{b}\rangle g\langle\boldsymbol{c}])$$

$$\mathrm{nfNmt}([\boldsymbol{a}+\boldsymbol{b}\rangle f \oplus g\langle\boldsymbol{c}+\boldsymbol{d}]) = \mathrm{nfNmt}([\boldsymbol{a}\rangle f\langle\boldsymbol{c}]) \uplus \mathrm{nfNmt}([\boldsymbol{b}\rangle g\langle\boldsymbol{d}])$$

$$\mathrm{nfNmt}(id_a) = id_a$$

$$\mathrm{nfNmt}(\delta_{ab}) = \delta_{ab}$$

$$\mathrm{nfNmt}(f \ ; \ g) = \mathrm{nfNmt}(f) \ ; \ \mathrm{nfNmt}(g)$$

$$\mathrm{nfNmt}(f \uplus g) = \mathrm{nfNmt}(f) \uplus \mathrm{nfNmt}(g)$$

$$\mathrm{nfNmt}(\pi \cdot f) = \pi \cdot \mathrm{nfNmt}(f)$$

Definition 7.44. We define Nmt : SMT → NMT as

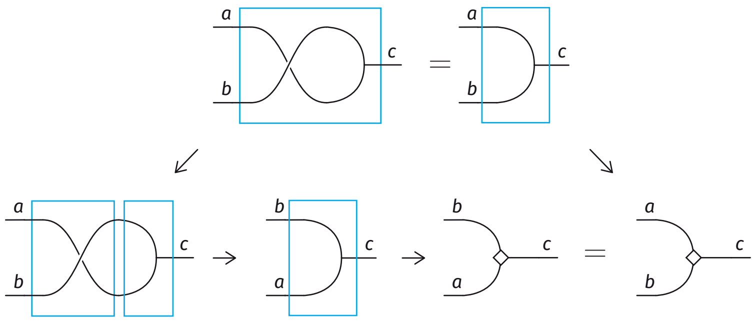
$$\mathrm{Nmt}\langle\Sigma, E\rangle = \langle\Sigma, \mathrm{nfNmt} \circ \mathrm{box}(E)\rangle$$

where we extend the function nfNmt on a set of equations in the obvious way:

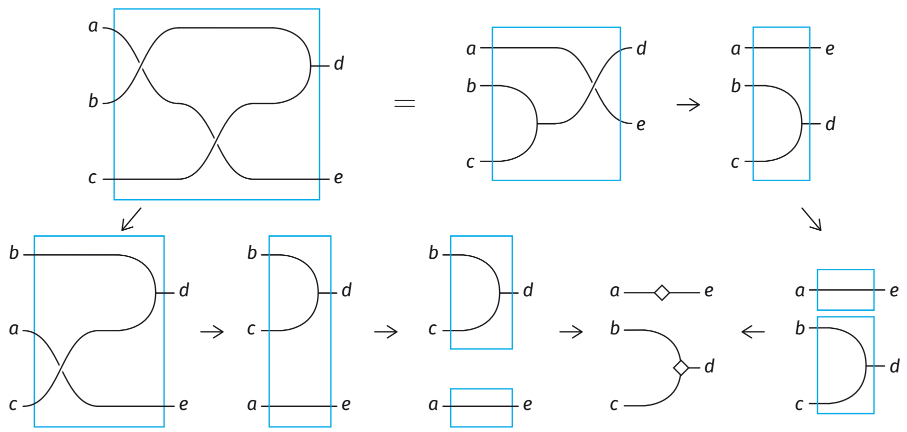
$$\mathrm{nfNmt}(E) = \{\mathrm{nfNmt}(f) = \mathrm{nfNmt}(g) \mid f = g \in E\}$$

We now return to fig. 7.3 and fig. 7.4 and show in the following example, that by applying Nmt to the equations in fig. 7.3 we obtain equations in fig. 7.4.

Example 7.45. In this example, we illustrate the translation of a rule of an SMT into the corresponding rule of an NMT via nfNmt. The diagram below shows the application of nfNmt to both sides of an equation in the SMT theory of surjections.



The next diagram illustrates the fact that the equation (SMT-nat) get subsumed by the equations of NMT, namely by (NMT-comm).



7.6.3 Completeness of NMTs

We now show that the constructions from the previous two sections yield the same nPROP. Starting from an SMT $\langle \Sigma, E\rangle$, we can either translate it into a PROP and then apply NOM, or we can first translate the SMT theory into an NMT theory via nfNmt and then turn it into an nPROP, as illustrated by fig. 7.10.

$$
\begin{tikzcd}
\mathrm{SMT} \arrow[rrr, "\mathrm{Prop}"] \arrow[dddd, "\mathrm{Nmt}"'] & & & \mathrm{PROP} \arrow[dddd, "\mathrm{NOM}"] \\
& \langle \Sigma, E\rangle \arrow[r, mapsto] \arrow[ddr, mapsto] & \mathrm{Prop}\,\langle \Sigma, E\rangle \arrow[d, mapsto] & \\
& & \mathrm{NOM}(\mathrm{Prop}\,\langle \Sigma, E\rangle) \arrow[d, bend left, "\mathrm{nfNmt}"] & \\
& & \mathrm{nProp}(\mathrm{Nmt}\,\langle \Sigma, E\rangle) \arrow[u, bend left, "\iota"] & \\
\mathrm{NMT} \arrow[rrr, "\mathrm{nProp}"] & & & \mathrm{nPROP}
\end{tikzcd}
$$

Figure 7.10: Completing the square

We set up some preliminaries. First, we define the map $\iota : n\mathrm{Trm}(\Sigma) \to n\mathrm{Trm}(\mathrm{Trm}(\Sigma))$, which is an injection map going in the opposite direction to nfNmt:

$$\underset{\sim}{\iota}([\boldsymbol{a}\rangle\gamma\langle\boldsymbol{b}]) = [\boldsymbol{a}\rangle\underline{\gamma}\langle\boldsymbol{b}] \text{ where } \gamma \in \Sigma$$

$$\underset{\sim}{\iota}(id_a) = id_a$$

$$\underset{\sim}{\iota}(\delta_{ab}) = \delta_{ab}$$

$$\underset{\sim}{\iota}(f \ ; \ g) = \underset{\sim}{\iota}(f) \ ; \ \underset{\sim}{\iota}(g)$$

$$\underset{\sim}{\iota}(f \uplus g) = \underset{\sim}{\iota}(f) \uplus \underset{\sim}{\iota}(g)$$

$$\underset{\sim}{\iota}(\pi \cdot f) = \pi \cdot \underset{\sim}{\iota}(f)$$

The only interesting thing happens in the case for a nominal generator, which gets turned into an ordinary string diagram, embedded in a nominal diagram.

Next, we show that the maps nfNmt and $\underset{\sim}{\iota}$ are inverses of each other (up to some equational reasoning).

Lemma 7.46. *We have* $\mathsf{nfNmt} \circ \underset{\sim}{\iota}\,(f) = f$ *for any* $f \in \mathsf{nTrm}(\Sigma)$.

Proof. By induction on f.

□

Lemma 7.47. *We have* $\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,(f) \overset{\mathrm{NOM}}{\underset{\sim}{=}} f$ *for any* $f \in \mathsf{nTrm}(\mathsf{Trm}(\Sigma))$*. Where* $\overset{\mathrm{NOM}}{\underset{\sim}{=}}$ *is equality up to the equations* $\mathbf{NOM} \cup \mathbf{NMT} \cup \mathsf{box}[\mathbf{SMT}]$.

Proof. By induction on f, we see that all the cases follow either by $\underset{\sim}{\iota} \circ \mathsf{nfNmt}$ being the identity, such as in the case of $f = s \uplus t$, s.t.

$$\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,(s \uplus t) = \underset{\sim}{\iota}(\mathsf{nfNmt}\,(s) \uplus \mathsf{nfNmt}\,(t)) \overset{\mathrm{NOM}}{\underset{\sim}{=}} (\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,(s)) \uplus (\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,(t)) \overset{\mathrm{NOM}}{\underset{\sim}{=}} s \uplus t$$

Where the last equality follows by the IH.

For almost all f of the shape $[\boldsymbol{a}\rangle s\langle\boldsymbol{b}]$, the proposition follows directly from the equations $\mathbf{NOM}$, for example, if $f = [\boldsymbol{a}\rangle s \ ; \ t\langle\boldsymbol{c}]$ we have

$$\begin{aligned}\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,([\boldsymbol{a}\rangle s \ ; \ t\langle\boldsymbol{c}]) &= (\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,([\boldsymbol{a}\rangle s\langle\boldsymbol{b}])) \ ; \ (\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,([\boldsymbol{b}\rangle t\langle\boldsymbol{c}])) \\ &\overset{\mathrm{NOM}}{\underset{\sim}{=}} [\boldsymbol{a}\rangle s\langle\boldsymbol{b}] \ ; \ [\boldsymbol{b}\rangle t\langle\boldsymbol{c}] \overset{\mathrm{NOM}}{\underset{\sim}{=}} [\boldsymbol{a}\rangle s \ ; \ t\langle\boldsymbol{c}]\end{aligned}$$

In the case of $f = [\boldsymbol{a}\rangle\underline{\gamma}\langle\boldsymbol{c}]$, we have

$$\underset{\sim}{\iota} \circ \mathsf{nfNmt}\,([\boldsymbol{a}\rangle\underline{\gamma}\langle\boldsymbol{c}]) = \underset{\sim}{\iota}\,([\boldsymbol{a}\rangle\gamma\langle\boldsymbol{c}]) = [\boldsymbol{a}\rangle\underline{\gamma}\langle\boldsymbol{c}]$$

The only case which requires further analysis is $f = [ab\rangle\sigma\langle cd]$, for which we have

$$
\begin{aligned}
\iota \circ \mathsf{nfNmt}\,([ab\rangle\sigma\langle cd]) &= \iota\,([ab|dc]) \\
&= [ab|dc] \overset{\mathsf{NOM}}{=} [ba|cd] \overset{\mathsf{NOM}}{=} [ba\rangle id\langle cd] \\
&\overset{\mathsf{NOM}}{=} [ab|ab] \mathbin{;} [ba\rangle id\langle cd] \\
&\overset{\mathsf{NOM}}{=} [ab\rangle\langle ab|ba\rangle \mathbin{;} id\langle cd] \\
&\overset{\mathsf{NOM}}{=} [ab\rangle\langle ab|ba\rangle\langle cd] \\
&= [ab\rangle\sigma\langle cd]
\end{aligned}
$$

□

Lemma 7.48. *The diagram in fig. 7.10 commutes*

Proof. We want to show that the two maps nfNmt and ι are isomorphisms. By definition, both nfNmt and ι are homomorphisms between the term algebras and we have shown in lem. 7.46 that $\mathsf{nfNmt} \circ \iota\,(f) = f$ and $\iota \circ \mathsf{nfNmt}\,(f) \overset{\mathsf{NOM}}{=} f$ follows from lem. 7.47.

To verify that these maps are well-defined, that is, maps between equivalence classes of $n\mathsf{Trm}$s, we need to check that they preserve the equations:

- for the map ι, we have to show

$$\iota[\mathcal{T}h(\mathsf{nfNmt}[\mathsf{box}[E]] \cup \mathbf{NMT})] \subseteq \mathcal{T}h(\mathcal{T}h(\mathsf{box}[E \cup \mathbf{SMT}]) \cup \mathbf{NOM} \cup \mathbf{NMT})$$

  In fact, by lem. 7.43, it suffices to check that $\iota[\mathsf{nfNmt}[\mathsf{box}[E]]] \subseteq \mathcal{T}h(\mathsf{box}[E] \cup \mathbf{NMT})$ and $\iota[\mathbf{NMT}] \subseteq \mathcal{T}h(\mathsf{box}[E] \cup \mathbf{NMT})$. The first inequality follows immediately from the fact that $\iota[\mathsf{nfNmt}[\mathsf{box}[E]]] = \mathsf{box}[E]$. The second inequality follows straightforwardly.

- for the map nfNmt, we have to show the other direction

$$\mathsf{nfNmt}[\mathcal{T}h(\mathcal{T}h(\mathsf{box}[E \cup \mathbf{SMT}]) \cup \mathbf{NOM} \cup \mathbf{NMT})] \subseteq \mathcal{T}h(\mathsf{nfNmt}[\mathsf{box}[E]] \cup \mathbf{NMT})$$

  By lem. 7.43, we have $\mathsf{nfNmt}[\mathcal{T}h(X)] \subseteq \mathcal{T}h(\mathsf{nfNmt}[X])$, in the following chain of inequalities:

$$
\begin{aligned}
& \mathrm{nfNmt}[\mathcal{Th}(\mathcal{Th}(\mathrm{box}[E \cup \mathbf{SMT}]) \cup \mathbf{NOM} \cup \mathbf{NMT})] \\
\subseteq\ & \mathcal{Th}(\mathrm{nfNmt}[\mathcal{Th}(\mathrm{box}[E \cup \mathbf{SMT}]) \cup \mathbf{NOM} \cup \mathbf{NMT}]) \\
=\ & \mathcal{Th}(\mathrm{nfNmt}[\mathcal{Th}(\mathrm{box}[E \cup \mathbf{SMT}])) \cup \mathrm{nfNmt}[\mathbf{NOM} \cup \mathbf{NMT}]) \\
\subseteq\ & \mathcal{Th}(\mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E \cup \mathbf{SMT}]]) \cup \mathrm{nfNmt}[\mathbf{NOM} \cup \mathbf{NMT}]) \\
=\ & \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E \cup \mathbf{SMT}]] \cup \mathrm{nfNmt}[\mathbf{NOM} \cup \mathbf{NMT}]) \\
=\ & \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathrm{nfNmt}[\mathrm{box}[\mathbf{SMT}]] \cup \mathrm{nfNmt}[\mathbf{NOM}] \cup \mathrm{nfNmt}[\mathbf{NMT}]) \\
\subseteq\ & \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathbf{NMT})
\end{aligned}
$$

To justify the last inequality, we only need to prove:

- $\mathrm{nfNmt}[\mathrm{box}[E]] \subseteq \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathbf{NMT})$

  Follows immediately.

- $\mathrm{nfNmt}[\mathrm{box}[\mathbf{SMT}]] \subseteq \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathbf{NMT})$

  The equations in **SMT** get subsumed by **NMT** when box-ed and normalised via nfNmt. We only show the most interesting equation (SMT-nat). For a graphical intuition of this equality, see ex. 7.45. We write $\boldsymbol{a}^m$ for a list of

a's of length m in the equational reasoning below:

$$
\begin{aligned}
& \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m + \boldsymbol{a}^z\rangle\,(t \oplus id_z)\ ;\ \sigma_{n,z}\,\langle \boldsymbol{b}^z + \boldsymbol{b}^n])\\
&= \underset{\sim\sim\sim}{\mathsf{nfNmt}}(\boldsymbol{a}^m + \boldsymbol{a}^z\rangle\, t \oplus id_z\,\langle \mathbf{x}^n + \mathbf{x}^z])\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{x}^n + \mathbf{x}^z\rangle\,\sigma_{n,z}\,\langle \boldsymbol{b}^z + \boldsymbol{b}^n])\\
&= (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m\rangle\, t\,\langle \mathbf{x}^n]) \uplus \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^z\rangle\, id_z\,\langle \mathbf{x}^z]))\ ;\ [\mathbf{x}^n + \mathbf{x}^z\,|\,\boldsymbol{b}^n + \boldsymbol{b}^z]\\
&= (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m\rangle\, t\,\langle \mathbf{x}^n]) \uplus [\boldsymbol{a}^z\,|\,\mathbf{x}^z])\ ;\ [\mathbf{x}^n + \mathbf{x}^z\,|\,\boldsymbol{b}^n + \boldsymbol{b}^z]\\
&\overset{\mathsf{NMT}}{=} (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m\rangle\, t\,\langle \mathbf{x}^n])\ ;\ [\mathbf{x}^n\,|\,\boldsymbol{b}^n]) \uplus ([\boldsymbol{a}^z\,|\,\mathbf{x}^z]\ ;\ [\mathbf{x}^z\,|\,\boldsymbol{b}^z])\\
&\overset{\mathsf{NMT}}{=} (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m\rangle\, t\,\langle \mathbf{x}^n])\ ;\ [\mathbf{x}^n\,|\,\boldsymbol{b}^n]) \uplus [\boldsymbol{a}^z\,|\,\boldsymbol{b}^z]\\
&\overset{\mathsf{NMT}^*}{=} \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m\rangle\, t\,\langle \boldsymbol{b}^n]) \uplus [\boldsymbol{a}^z\,|\,\boldsymbol{b}^z]\\
&\overset{\mathsf{NMT}^*}{=} ([\boldsymbol{a}^m\,|\,\mathbf{y}^m]\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^m\rangle\, t\,\langle \boldsymbol{b}^n])) \uplus [\boldsymbol{a}^z\,|\,\boldsymbol{b}^z]\\
&\overset{\mathsf{NMT}}{=} ([\boldsymbol{a}^m\,|\,\mathbf{y}^m]\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^m\rangle\, t\,\langle \boldsymbol{b}^n])) \uplus ([\boldsymbol{a}^z\,|\,\mathbf{y}^z]\ ;\ [\mathbf{y}^z\,|\,\boldsymbol{b}^z])\\
&\overset{\mathsf{NMT}}{=} [\boldsymbol{a}^m + \boldsymbol{a}^z\,|\,\mathbf{y}^m + \mathbf{y}^z]\ ;\ (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^m\rangle\, t\,\langle \boldsymbol{b}^n]) \uplus [\mathbf{y}^z\,|\,\boldsymbol{b}^z])\\
&\overset{\mathsf{NMT}}{=} [\boldsymbol{a}^m + \boldsymbol{a}^z\,|\,\mathbf{y}^m + \mathbf{y}^z]\ ;\ ([\mathbf{y}^z\,|\,\boldsymbol{b}^z] \uplus \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^m\rangle\, t\,\langle \boldsymbol{b}^n]))\\
&= [\boldsymbol{a}^m + \boldsymbol{a}^z\,|\,\mathbf{y}^m + \mathbf{y}^z]\ ;\ (\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^z\rangle\, id_z\,\langle \boldsymbol{b}^z]) \uplus \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^m\rangle\, t\,\langle \boldsymbol{b}^n]))\\
&= [\boldsymbol{a}^m + \boldsymbol{a}^z\,|\,\mathbf{y}^m + \mathbf{y}^z]\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^z + \mathbf{y}^m\rangle\, id_z \oplus t\,\langle \boldsymbol{b}^z + \boldsymbol{b}^n])\\
&= \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m + \boldsymbol{a}^z\rangle\,\sigma_{m,z}\,\langle \mathbf{y}^z + \mathbf{y}^m])\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{y}^z + \mathbf{y}^m\rangle\, id_z \oplus t\,\langle \boldsymbol{b}^z + \boldsymbol{b}^n])\\
&= \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}^m + \boldsymbol{a}^z\rangle\,\sigma_{m,z}\ ;\ (id_z \oplus t)\,\langle \boldsymbol{b}^z + \boldsymbol{b}^n])
\end{aligned}
$$

To justify the two steps in the middle of the derivation (marked with $\star$), we need to show:

$$[\boldsymbol{a}\,|\,\mathbf{x}]\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{x}\rangle\, t\,\langle \boldsymbol{b}]) \overset{\mathsf{NMT}}{=} \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}\rangle\, t\,\langle \boldsymbol{b}]) \overset{\mathsf{NMT}}{=} \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}\rangle\, t\,\langle \mathbf{y}])\ ;\ [\mathbf{y}\,|\,\boldsymbol{b}]$$

This follows straightforwardly by induction on t:

- If $t = \gamma$ where $\gamma \in \Sigma$, then we can apply (NMT-left) for all $\delta_{a^i x^i}$s to get the LHS equality and (NMT-right) for all $\delta_{y^i b^i}$s to get the RHS.
- If $t = id$, then $\underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{x}\rangle\, t\,\langle \boldsymbol{b}]) = [\mathbf{x}\,|\,\boldsymbol{b}]$ and both equalities follow by the second equation of fig. 7.8.
- If $t = p\ ;\ q$, by IH $[\boldsymbol{a}\,|\,\mathbf{x}]\ ;\ \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\mathbf{x}\rangle\, p\,\langle \mathbf{z}]) \overset{\mathsf{NMT}}{=} \underset{\sim\sim\sim}{\mathsf{nfNmt}}([\boldsymbol{a}\rangle\, p\,\langle \mathbf{z}])$, thus we

have:

$$
\begin{aligned}
&[\boldsymbol{a}|\mathbf{x}] \mathbin{;} \mathrm{nfNmt}([\mathbf{x}\rangle\, p \mathbin{;} q\, \langle \boldsymbol{b}]) \\
&= [\boldsymbol{a}|\mathbf{x}] \mathbin{;} \mathrm{nfNmt}([\mathbf{x}\rangle p \langle \mathbf{z}]) \mathbin{;} \mathrm{nfNmt}([\mathbf{z}\rangle q \langle \boldsymbol{b}]) \\
&\overset{\text{NMT}}{=} \mathrm{nfNmt}([\boldsymbol{a}\rangle\, p\, \langle \mathbf{z}]) \mathbin{;} \mathrm{nfNmt}([\mathbf{z}\rangle\, q\, \langle \boldsymbol{b}]) \\
&= \mathrm{nfNmt}([\boldsymbol{a}\rangle\, t\, \langle \boldsymbol{b}])
\end{aligned}
$$

The RHS equality follows in a similar fashion.

* If $t = p \oplus q$, we reason as in the case of $t = p \mathbin{;} q$.

- $\mathrm{nfNmt}[\mathbf{NOM}] \subseteq \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathbf{NMT})$

  The only two equations which require any serious verification are (NOM-4) and (NOM-5). The proofs of both are essentially the same, so we will only consider the first one here:

$$
\begin{aligned}
\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle \mathbin{;} f\langle \mathbf{c}]) &= \mathrm{nfNmt}([\boldsymbol{a}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \mathbf{x}]) \mathbin{;} \mathrm{nfNmt}([\mathbf{x}\rangle f\langle \mathbf{c}]) \\
&\overset{\text{NMT}}{=} \mathrm{nfNmt}([\boldsymbol{a}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{b}']) \mathbin{;} \mathrm{nfNmt}([\boldsymbol{b}'\rangle f\langle \mathbf{c}]) \\
&\overset{\text{NMT}}{=} [\boldsymbol{a}|\boldsymbol{b}] \mathbin{;} [\boldsymbol{b}|\boldsymbol{a}] \mathbin{;} \mathrm{nfNmt}([\boldsymbol{a}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{b}']) \mathbin{;} \mathrm{nfNmt}([\boldsymbol{b}'\rangle f\langle \mathbf{c}]) \\
&\overset{\text{NMT}}{=} [\boldsymbol{a}|\boldsymbol{b}] \mathbin{;} \mathrm{nfNmt}([\boldsymbol{b}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{b}']) \mathbin{;} \mathrm{nfNmt}([\boldsymbol{b}'\rangle f\langle \mathbf{c}]) \\
&\overset{\text{NMT}}{=} [\boldsymbol{a}|\boldsymbol{b}] \mathbin{;} \mathrm{nfNmt}([\boldsymbol{b}'\rangle f\langle \mathbf{c}])
\end{aligned}
$$

  For the justification of $\mathrm{nfNmt}([\boldsymbol{b}\rangle\langle \boldsymbol{b}|\boldsymbol{b}'\rangle\langle \boldsymbol{b}']) \overset{\text{NMT}}{=} id$ see the remark below.

- $\mathrm{nfNmt}[\mathbf{NMT}] \subseteq \mathcal{Th}(\mathrm{nfNmt}[\mathrm{box}[E]] \cup \mathbf{NMT})$

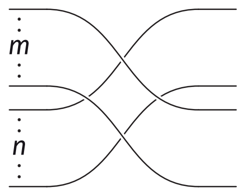
  Follows straightforwardly.

□

Remark 7.49. Whilst we have been using the $\langle -|-\rangle$ notation as syntactic sugar for arbitrary bijections (in PROP) throughout the last two chapters, we have not provided a rigorous definition beyond the informal description at the beginning of sec. 7.5. Below we give an inductive definition for this notation and prove that

$$
\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle \boldsymbol{a}|\boldsymbol{a}'\rangle\langle \boldsymbol{a}]) \overset{\text{NMT}}{=} id_A
$$

First we review some preliminaries. We write $\sigma_{m,n} : m \oplus n \to n \oplus m$ for the diagram



In the following definition we will write $\boldsymbol{a}\,!\,x$ for the indexing function, which, given a list $\boldsymbol{a}$ and an element x, returns the position (index) of the element in the list. Finally, we denote the underlying set of $\boldsymbol{a}$ by A and $|\boldsymbol{a}|$ stands for the length of $\boldsymbol{a}$.

$$
\begin{aligned}
&\langle - | - \rangle : (\boldsymbol{a} : \Sigma\, list) \times (\boldsymbol{a}' : \Sigma\, list) \to |\boldsymbol{a}| \to |\boldsymbol{a}'| \\
&\langle [] | \boldsymbol{a} \rangle = id_{|\boldsymbol{a}|} \\
&\langle x : xs | \boldsymbol{a} \rangle = \begin{cases} (id_i \oplus \sigma_{1,j-i} \oplus id_{|\boldsymbol{a}|-(j+1)}) \,;\, \langle xs | \boldsymbol{a} \rangle & \text{if } i < j \\ \langle xs | \boldsymbol{a} \rangle & \text{if } i = j \\ (id_j \oplus \sigma_{i-j,1} \oplus id_{|\boldsymbol{a}|-(i+1)}) \,;\, \langle xs | \boldsymbol{a} \rangle & \text{otherwise} \end{cases} \\
&\qquad\qquad\quad \textbf{where } i = |\boldsymbol{a}| - |x : xs| \\
&\qquad\qquad\qquad\qquad j = \boldsymbol{a}\,!\,x
\end{aligned}
$$

Now we show

$$\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle xs|\boldsymbol{a}'\rangle\langle\boldsymbol{a}']) \overset{\mathrm{NMT}}{=} id_A$$

provided that $\boldsymbol{set}(xs) \subseteq A$, $\boldsymbol{set}(\boldsymbol{a}) = \boldsymbol{set}(\boldsymbol{a}')$ and $|\boldsymbol{a}| = |\boldsymbol{a}'|$.

Proof by induction on xs:

- If $xs = []$, then

$$\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle[]|\boldsymbol{a}'\rangle\langle\boldsymbol{a}]) = \mathrm{nfNmt}([\boldsymbol{a}\rangle id_{|\boldsymbol{a}'|}\langle\boldsymbol{a}]) = id_A$$

- If $xs = y : ys$, we have three cases:

- If $i < j$, then:

$$
\begin{aligned}
&\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle y : ys|\boldsymbol{a}'\rangle\langle\boldsymbol{a}'])\\
&= \mathrm{nfNmt}([\boldsymbol{a}\rangle\langle id_i \oplus \sigma_{1,j-i} \oplus id_{|\boldsymbol{a}'|-(j+1)}\rangle \mathbin{;} \langle ys|\boldsymbol{a}'\rangle\langle\boldsymbol{a}'])\\
&= \mathrm{nfNmt}([\boldsymbol{a}\rangle id_i \oplus \sigma_{1,j-i} \oplus id_{|\boldsymbol{a}'|-(j+1)}\langle\mathbf{x}]) \mathbin{;} \mathrm{nfNmt}([\mathbf{x}\rangle\langle ys|\boldsymbol{a}\rangle\langle\boldsymbol{a}'])\\
&\overset{\mathrm{NMT}}{=} \mathrm{nfNmt}([\boldsymbol{a}\rangle id_i \oplus \sigma_{1,j-i} \oplus id_{|\boldsymbol{a}'|-(j+1)}\langle\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}]) \mathbin{;}\\
&\quad \mathrm{nfNmt}([\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}\rangle\langle ys|\boldsymbol{a}\rangle\langle\boldsymbol{a}'])\\
&= (\mathrm{nfNmt}([\boldsymbol{a}_{0\ldots i-1}\rangle id_i\langle\boldsymbol{a}_{0\ldots i-1}]) \uplus\\
&\quad \mathrm{nfNmt}([\boldsymbol{a}_{i\ldots j}\rangle\sigma_{1,j-i}\langle\boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i]) \uplus\\
&\quad \mathrm{nfNmt}([\boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}\rangle id_{|\boldsymbol{a}'|-(j+1)}\langle\boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}])) \mathbin{;}\\
&\quad \mathrm{nfNmt}([\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}\rangle\langle ys|\boldsymbol{a}\rangle\langle\boldsymbol{a}'])\\
&= ([\boldsymbol{a}_{0\ldots i-1}|\langle\boldsymbol{a}_{0\ldots i-1}] \uplus [\boldsymbol{a}_{i\ldots j}|\boldsymbol{a}_{i\ldots j}] \uplus [\boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}|\boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}]) \mathbin{;}\\
&\quad \mathrm{nfNmt}([\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}\rangle\langle ys|\boldsymbol{a}\rangle\langle\boldsymbol{a}'])\\
&\overset{\mathrm{NMT}}{=} \mathrm{nfNmt}([\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}\rangle\langle ys|\boldsymbol{a}\rangle\langle\boldsymbol{a}']) \overset{\mathrm{NMT}}{=} id_A
\end{aligned}
$$

Note that the last equality follows by IH, since we have $\mathit{set}(ys) \subseteq A$, $\mathit{set}(\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}) = \mathit{set}(\boldsymbol{a}) = \mathit{set}(\boldsymbol{a}')$ and $|\boldsymbol{a}_{0\ldots i-1} + \boldsymbol{a}_{i+1\ldots j} + \boldsymbol{a}_i + \boldsymbol{a}_{j+1\ldots|\boldsymbol{a}|-1}| = |\boldsymbol{a}| = |\boldsymbol{a}'|$.

- If $i = j$ then we immediately have

$$\mathrm{nfNmt}([\boldsymbol{a}\rangle\langle y : ys|\boldsymbol{a}'\rangle\langle\boldsymbol{a}']) = \mathrm{nfNmt}([\boldsymbol{a}\rangle\langle ys|\boldsymbol{a}'\rangle\langle\boldsymbol{a}']) \overset{\mathrm{NMT}}{=} id_A$$

- If $i > j$ then the reasoning is symmetric to the case $i < j$,

In sec. 7.3, we introduced the NMT theories for the categories of bijections, injections, surjections and functions on names (amongst others). In thm. 7.20, we stated that the theories presented in fig. 7.4 are sound and complete. Below we restate this result, now with a proof.

Theorem 7.50. *[Completeness of NMTs]*

The calculi of fig. 7.4 are sound and complete, that is, the categories presented by these calculi are isomorphic to the categories of finite sets of names with the respective maps.

Proof. We show the result for the category of finite functions $\mathrm{n}\mathbb{F}$. Similar arguments apply to the other theories presented in fig. 7.4. First, by completeness of an SMT $\langle\Sigma, E\rangle$ with regards to some category $\mathbb{C}$, we mean that the PROP presented by $\langle\Sigma, E\rangle$ is isomorphic to $\mathbb{C}$

$$\mathrm{Prop}\,\langle \Sigma, E\rangle \cong \mathbb{C}$$

Likewise, we define the completeness for an NMT with regards to some category $\mathrm{n}\mathbb{C}$, as

$$\mathrm{nProp}\,\langle \Sigma, E\rangle \cong \mathrm{n}\mathbb{C}$$

In order to show completeness of the nominal theory of functions w.r.t. $\mathrm{n}\mathbb{F}$, we start with the SMT $\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle$ (see fig. 7.3).

From lem. 7.48 we know that

$$\mathrm{NOM}(\mathrm{Prop}\,\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle) \cong \mathrm{nProp}(\mathrm{Nmt}\,\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle)$$

From ex. 7.27 we know that

$$\mathrm{NOM}(\mathbb{F}) \cong \mathrm{n}\mathbb{F}$$

From completeness of $\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle$ for $\mathbb{F}$ we know

$$\mathrm{Prop}\,\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle \cong \mathbb{F}$$

Putting these together, we obtain

$$\mathrm{nProp}(\mathrm{Nmt}\,\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle) \cong \mathrm{n}\mathbb{F}$$

that is, $\mathrm{Nmt}\,\langle \Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle$ is complete for $\mathrm{n}\mathbb{F}$.

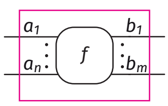
□

7.6.4 Embedding nominal PROPSs into PROPSs

In order to go back from nominal to ordinary string diagrams, we can build a similar construction to the one in sec. 7.6.1, by taking an NMT $\langle \Sigma, E\rangle$ to $\langle \mathrm{Trm}(\mathrm{dia}(n\mathrm{Trm}(\Sigma))), \mathrm{dia}(E) \cup \mathbf{ORD}\rangle$, where:

- $\mathrm{dia}(t : A \to B) = \langle \boldsymbol{a}]\, t\, [\boldsymbol{b}\rangle$ where $\mathit{set}(\boldsymbol{a}) = A$ and $\mathit{set}(\boldsymbol{b}) = B$, which is extended to a set of equations in the obvious way $\mathrm{dia}(E) = \{\langle \boldsymbol{a}]\, s\, [\boldsymbol{b}\rangle = \langle \boldsymbol{a}]\, t\, [\boldsymbol{b}\rangle \mid s = t \in E\}$
- **ORD** are the equations from prop. 7.29

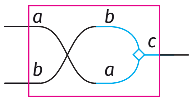
We draw $\langle \boldsymbol{a}]\, f\, [\boldsymbol{b}\rangle$ as



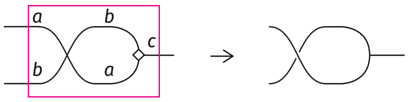
7.6.5 Translating NMTs into SMTs

This section follows the same lines as sec. 7.6.2, but now translating nominal monoidal theories into symmetric monoidal theories. Indeed, we can convert an NMT into an SMT by first embedding nominal equations into ordinary string diagrams and then normalising the diagrams via a function nfSmt, which we are going to define now.

Compared to normalising ordinary string diagrams embedded in the nominal setting, normalising embedded nominal string diagrams into ordinary string diagrams is slightly more tricky. This is due to the fact that in nominal sequential composition, we are allowed to compose two diagrams which share the same set of output and input labels, disregarding the order of the named ports. For example, in the picture below, we see a wire crossing inside the purple box, introduced by the fact that the ports of the box interface and the ports of the generator inside the box have to be lined up.



However, no such crossing is (directly) visible in the linear syntax $\langle a, b][b, a\rangle\mu\langle c][c\rangle$. Thus, when translating such a diagram back into an ordinary string diagram, we might need to insert some symmetries, i.e. the diagram $\langle a, b][b, a\rangle\mu\langle c][c\rangle$ should normalise to $\sigma \,;\, \mu$:



After these preliminary considerations, we now define nfSmt : Trm(dia(nTrm(Σ))) → Trm(Σ):

$$\mathrm{nfSmt}\,(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\gamma\langle \boldsymbol{b}'][\boldsymbol{b}\rangle) = \langle \boldsymbol{a}|\boldsymbol{a}'\rangle \,;\, \underline{\gamma} \,;\, \langle \boldsymbol{b}'|\boldsymbol{b}\rangle \text{ where } \gamma \in \Sigma$$

$$\mathrm{nfSmt}\,(\langle a]id_a[a\rangle) = id$$

$$\mathrm{nfSmt}\,(\langle a]\delta_{ab}[b\rangle) = id$$

$$\mathrm{nfSmt}\,(\langle \boldsymbol{a}]f \,;\, g[\boldsymbol{c}\rangle) = \mathrm{nfSmt}\,(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) \,;\, \mathrm{nfSmt}\,(\langle \boldsymbol{b}]g[\boldsymbol{c}\rangle)$$

$$\mathrm{nfSmt}\,(\langle \boldsymbol{a}]f \uplus g[\boldsymbol{b}\rangle) = \langle \boldsymbol{a}|\boldsymbol{a}_1 + \boldsymbol{a}_2\rangle \,;\, (\mathrm{nfSmt}\,(\langle \boldsymbol{a}_1]f[\boldsymbol{b}_1\rangle) \oplus \mathrm{nfSmt}\,(\langle \boldsymbol{a}_2]g[\boldsymbol{b}_2\rangle)) \,;\, \langle \boldsymbol{b}_1 + \boldsymbol{b}_2|\boldsymbol{b}\rangle$$

$$\mathrm{nfSmt}\,(\langle \boldsymbol{a}]\pi \cdot f[\boldsymbol{b}\rangle) = \mathrm{nfSmt}\,(\langle \pi^{-1} \cdot \boldsymbol{a}]\, f\, [\pi^{-1} \cdot \boldsymbol{b}\rangle)$$

$$\mathrm{nfSmt}\,(\gamma) = \gamma \text{ where } \gamma \in \Sigma$$

$$\mathrm{nfSmt}\,(id) = id$$

$$\mathrm{nfSmt}\,(\sigma) = \sigma$$

$$\mathrm{nfSmt}\,(f \,;\, g) = \mathrm{nfSmt}\,(f) \,;\, \mathrm{nfSmt}\,(g)$$

$$\mathrm{nfSmt}\,(f \oplus g) = \mathrm{nfSmt}\,(f) \oplus \mathrm{nfSmt}\,(g)$$

Definition 7.51. We define Smt : NMT → SMT as

$$\mathsf{Smt}\,\langle \Sigma, E\rangle = \langle \Sigma, \mathsf{nfSmt} \circ \mathsf{dia}(E)\rangle$$

7.6.6 Completeness of SMTs

We now show that the constructions from the previous two sections yield the same PROP, namely, starting from an NMT $\langle \Sigma, E\rangle$, we can either translate it into a nPROP and then apply ORD, or we can first translate the NMT theory into an SMT theory via nfSmt and then turn it into a PROP.

$$\begin{tikzcd}
\mathsf{NMT} \arrow[rrr, "\mathsf{nProp}"] \arrow[dddd, "\mathsf{Smt}"'] & & & \mathsf{nPROP} \arrow[dddd, "\mathsf{ORD}"] \\
& \langle \Sigma, E\rangle \arrow[r, mapsto] \arrow[ddr, mapsto] & \mathsf{nProp}\,\langle \Sigma, E\rangle \arrow[d, mapsto] & \\
& & \mathsf{ORD}(\mathsf{nProp}\,\langle \Sigma, E\rangle) \arrow[d, bend left, "\mathsf{nfSmt}"] & \\
& & \mathsf{Prop}(\mathsf{Smt}\,\langle \Sigma, E\rangle) \arrow[u, bend left, "\iota"] & \\
\mathsf{SMT} \arrow[rrr, "\mathsf{Prop}"] & & & \mathsf{PROP}
\end{tikzcd}$$

Figure 7.11: Completing the square

We set up some preliminaries. First, we define the map $\iota : \mathsf{Trm}(\Sigma) \to \mathsf{Trm}(\mathsf{dia}(n\mathsf{Trm}(\Sigma)))$, which is an injection map going in the opposite direction to nfSmt:

$$\iota(\gamma) = \langle \boldsymbol{a}][\boldsymbol{a}\rangle\gamma\langle \boldsymbol{b}][\boldsymbol{b}\rangle \text{ where } \gamma \in \Sigma$$

$$\iota(id) = id$$

$$\iota(\sigma) = \sigma$$

$$\iota(f \,;\, g) = \iota(f) \,;\, \iota(g)$$

$$\iota(f \oplus g) = \iota(f) \oplus \iota(g)$$

Next, we show that the maps nfSmt and ι are inverses of each other.

Lemma 7.52. *We have* $\mathsf{nfSmt} \circ \iota\,(f) \overset{\mathsf{SMT}}{=} f$ *for any* $f \in \mathsf{Trm}(\Sigma)$.

Proof. By induction on f. The only case of interest is $f = \underline{\gamma}$ where $\gamma \in \Sigma$:

$$\mathsf{nfSmt} \circ \iota\,(\underline{\gamma}) = \mathsf{nfSmt}\,(\langle \boldsymbol{a}][\boldsymbol{a} \rangle \gamma \langle \boldsymbol{b}][\boldsymbol{b} \rangle) = \langle \boldsymbol{a} | \boldsymbol{a} \rangle \,;\, \underline{\gamma} \,;\, \langle \boldsymbol{b} | \boldsymbol{b} \rangle \overset{\mathsf{SMT}}{=} \underline{\gamma}$$

□

Lemma 7.53. *We have* $\iota \circ \mathsf{nfSmt}\,(f) \overset{\mathsf{ORD}}{\underset{\sim}{=}} f$ *for any* $f \in \mathsf{Trm}(\mathsf{dia}(n\mathsf{Trm}(\Sigma)))$. *Where* $\overset{\mathsf{ORD}}{\underset{\sim}{=}}$ *is equality up to the equations* $\mathbf{ORD} \cup \mathbf{SMT} \cup \mathsf{dia}[\mathbf{NMT}]$.

Proof. By induction on f:

- If $f = \langle \boldsymbol{a}][\boldsymbol{a}' \rangle \gamma \langle \boldsymbol{b}'][\boldsymbol{b} \rangle$, then

$$\begin{aligned}
\iota \circ \mathsf{nfSmt}\,(\langle \boldsymbol{a}][\boldsymbol{a}' \rangle \gamma \langle \boldsymbol{b}'][\boldsymbol{b} \rangle) &= \iota\,(\langle \boldsymbol{a} | \boldsymbol{a}' \rangle \,;\, \underline{\gamma} \,;\, \langle \boldsymbol{b}' | \boldsymbol{b} \rangle) \\
&= \langle \boldsymbol{a} | \boldsymbol{a}' \rangle \,;\, \langle \mathbf{x}][\mathbf{x} \rangle \gamma \langle \mathbf{y}][\mathbf{y} \rangle \,;\, \langle \boldsymbol{b}' | \boldsymbol{b} \rangle \\
&\overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}][\boldsymbol{a}' | \mathbf{x}] \,;\, [\mathbf{x} \rangle \gamma \langle \mathbf{y}] \,;\, [\mathbf{y} | \boldsymbol{b}'][\boldsymbol{b} \rangle \\
&\overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}][\boldsymbol{a}' \rangle \gamma \langle \boldsymbol{b}'][\boldsymbol{b} \rangle
\end{aligned}$$

  In the equational reasoning above, we use (ORD-4) and (ORD-5) in the third equality and then use the “lifted” rules (NMT-left) and (NMT-right) repeatedly to obtain the last equality.

- If $f = \langle a] id_a [a \rangle$, then

$$\iota \circ \mathsf{nfSmt}\,(\langle a] id_a [a \rangle) = \iota\,(id) = id \overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle a] id_a [a \rangle$$

- If $f = \langle a] \delta_{ab} [b \rangle$, then

$$\begin{aligned}
\iota \circ \mathsf{nfSmt}\,(\langle a] \delta_{ab} [b \rangle) &= \iota\,(id) = id \\
&\overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle a] id_a [a \rangle \\
&\overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle a][a | b] \,;\, [b | a][a \rangle \\
&\overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle a][a | b][b \rangle \,;\, \langle a | a \rangle \overset{\mathsf{ORD}}{\underset{\sim}{=}} \langle a] \delta_{ab} [b \rangle
\end{aligned}$$

- If $f = \langle \boldsymbol{a}]f \;;\; g[\boldsymbol{c}\rangle$, then

$$
\begin{aligned}
\iota \circ \mathrm{nfSmt}\,(\langle \boldsymbol{a}]f \;;\; g[\boldsymbol{c}\rangle) &= \iota\,(\mathrm{nfSmt}(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle) \;;\; \mathrm{nfSmt}(\langle \boldsymbol{b}]g[\boldsymbol{c}\rangle)) \\
&= \iota\,(\mathrm{nfSmt}(\langle \boldsymbol{a}]f[\boldsymbol{b}\rangle)) \;;\; \iota\,(\mathrm{nfSmt}(\langle \boldsymbol{b}]g[\boldsymbol{c}\rangle)) \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}]f[\boldsymbol{b}\rangle \;;\; \langle \boldsymbol{b}]g[\boldsymbol{c}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}]f \;;\; g[\boldsymbol{c}\rangle
\end{aligned}
$$

- If $f = \langle \boldsymbol{a}]f \uplus g[\boldsymbol{b}\rangle$, then

$$
\begin{aligned}
&\iota \circ \mathrm{nfSmt}\,(\langle \boldsymbol{a}]f \uplus g[\boldsymbol{b}\rangle) \\
&= \iota\,(\langle \boldsymbol{a}|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \;;\; (\mathrm{nfSmt}(\langle \boldsymbol{a}_1]f[\boldsymbol{b}_1\rangle) \oplus \mathrm{nfSmt}(\langle \boldsymbol{a}_2]g[\boldsymbol{b}_2\rangle)) \;;\; \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}\rangle) \\
&= \langle \boldsymbol{a}|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \;;\; \iota\,(\mathrm{nfSmt}(\langle \boldsymbol{a}_1]f[\boldsymbol{b}_1\rangle) \oplus \mathrm{nfSmt}(\langle \boldsymbol{a}_2]g[\boldsymbol{b}_2\rangle)) \;;\; \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}\rangle \\
&= \langle \boldsymbol{a}|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \;;\; (\iota\,(\mathrm{nfSmt}(\langle \boldsymbol{a}_1]f[\boldsymbol{b}_1\rangle)) \oplus \iota\,(\mathrm{nfSmt}(\langle \boldsymbol{a}_2]g[\boldsymbol{b}_2\rangle))) \;;\; \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \;;\; (\langle \boldsymbol{a}_1]f[\boldsymbol{b}_1\rangle \oplus \langle \boldsymbol{a}_2]g[\boldsymbol{b}_2\rangle) \;;\; \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \;;\; \langle \boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2]f \uplus g[\boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2\rangle \;;\; \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}][\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2|\boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2] \;;\; (f \uplus g) \;;\; [\boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2|\boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2][\boldsymbol{b}\rangle \overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}]f \uplus g[\boldsymbol{b}\rangle
\end{aligned}
$$

- If $f = \langle \boldsymbol{a}]\pi \cdot f[\boldsymbol{b}\rangle$, then

$$
\begin{aligned}
\iota \circ \mathrm{nfSmt}\,(\langle \boldsymbol{a}]\pi \cdot f[\boldsymbol{b}\rangle) &= \iota\,(\mathrm{nfSmt}(\langle \pi^{-1} \cdot \boldsymbol{a}]f[\pi^{-1} \cdot \boldsymbol{b}\rangle)) \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \pi^{-1} \cdot \boldsymbol{a}]f[\pi^{-1} \cdot \boldsymbol{b}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \pi \cdot \pi^{-1} \cdot \boldsymbol{a}]\pi \cdot f[\pi \cdot \pi - 1 \cdot \boldsymbol{b}\rangle \\
&\overset{\mathrm{ORD}}{\underset{\sim}{=}} \langle \boldsymbol{a}]\pi \cdot f[\boldsymbol{b}\rangle
\end{aligned}
$$

All the other cases of f follow straightforwardly (for the other cases, see the definition of nfSmt).

□

Lemma 7.54. *The diagram in fig. 7.11 commutes*

Proof. We want to show that the two maps nfSmt and ι are isomorphisms. By definition, both nfSmt and ι are homomorphisms between the term algebras and we have shown in lem. 7.52 that $\mathsf{nfSmt} \circ \iota\,(f) \overset{\mathsf{SMT}}{=} f$ and $\iota \circ \mathsf{nfSmt}\,(f) \overset{\mathsf{ORD}}{=} f$ follows from lem. 7.53.

To verify that these maps are well-defined, that is, maps between equivalence classes of Trms, we need to check that they preserve the equations:

- for the map ι, we have to show

$$\iota[\mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})] \subseteq \mathcal{Th}(\mathcal{Th}(\mathsf{dia}[E \cup \mathbf{NMT}]) \cup \mathbf{ORD} \cup \mathbf{SMT})$$

  In fact, by lem. 7.43, it suffices to check that $\iota[\mathsf{nfSmt}[\mathsf{dia}[E]]] \subseteq \mathcal{Th}(\mathsf{dia}[E \cup \mathsf{dia}[\mathbf{NMT}] \cup \mathbf{ORD} \cup \mathbf{SMT})$ and $\iota[\mathbf{SMT}] \subseteq \mathcal{Th}(\mathbf{SMT})$. The first inequality follows immediately from the fact that $\iota[\mathsf{nfSmt}[\mathsf{dia}[E]] \overset{\mathsf{ORD}}{=} \mathsf{dia}[E]$. The second inequality follows immediately.

- for the map nfSmt, we have to show the other direction

$$\mathsf{nfSmt}[\mathcal{Th}(\mathcal{Th}(\mathsf{dia}[E \cup \mathbf{NMT}]) \cup \mathbf{ORD} \cup \mathbf{SMT})] \subseteq \mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})$$

  We have :

$$\begin{aligned}
&\mathsf{nfSmt}[\mathcal{Th}(\mathcal{Th}(\mathsf{dia}[E \cup \mathbf{NMT}]) \cup \mathbf{ORD} \cup \mathbf{SMT})] \\
\subseteq\ &\mathcal{Th}(\mathsf{nfNmt}[\mathcal{Th}(\mathsf{dia}[E \cup \mathbf{NMT}]) \cup \mathbf{ORD} \cup \mathbf{SMT}]) \\
=\ &\mathcal{Th}(\mathsf{nfNmt}[\mathcal{Th}(\mathsf{dia}[E \cup \mathbf{NMT}])) \cup \mathsf{nfNmt}[\mathbf{ORD} \cup \mathbf{SMT}]) \\
\subseteq\ &\mathcal{Th}(\mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E \cup \mathbf{NMT}]]) \cup \mathsf{nfSmt}[\mathbf{ORD} \cup \mathbf{SMT}]) \\
=\ &\mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E \cup \mathbf{NMT}]] \cup \mathsf{nfSmt}[\mathbf{ORD} \cup \mathbf{SMT}]) \\
=\ &\mathcal{Th}(\mathsf{nfNmt}[\mathsf{dia}[E]] \cup \mathsf{nfSmt}[\mathsf{dia}[\mathbf{NMT}]] \cup \mathsf{nfSmt}[\mathbf{ORD}] \cup \mathsf{nfSmt}[\mathbf{SMT}]) \\
\subseteq\ &\mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})
\end{aligned}$$

  To justify the last inequality, we only need to prove:

  - $\mathsf{nfSmt}[\mathsf{dia}[E]] \subseteq \mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})$

    Follows immediately.

  - $\mathsf{nfSmt}[\mathsf{dia}[\mathbf{NMT}]] \subseteq \mathcal{Th}(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})$

    It is easy enough to see for most equations of $\mathsf{nfSmt}[\mathsf{dia}[\mathbf{NMT}]]$ are in

$\mathcal{T}\!h(\mathbf{SMT})$. For the interesting case of (NMT-comm) being preserved by nfSmt ∘ dia, see the proof in prop. 7.29.

- $\mathrm{nfSmt}[\mathbf{ORD}] \subseteq \mathcal{T}\!h(\mathrm{nfSmt}[\mathrm{dia}[E]] \cup \mathbf{SMT})$

  The only two equations which require any serious verification are (ORD-4) and (ORD-5). The proofs of both are essentially the same, so we will only consider the first one here:

$$\begin{aligned}
\mathrm{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}'|\boldsymbol{b}]; f[\boldsymbol{c}\rangle) &= \mathrm{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}'|\boldsymbol{b}][\boldsymbol{b}'\rangle) \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{b}']f[\boldsymbol{c}\rangle) \\
&\overset{\mathrm{SMT}}{=} \mathrm{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}'|\boldsymbol{b}][\boldsymbol{b}\rangle) \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{b}]f[\boldsymbol{c}\rangle) \\
&\overset{\mathrm{SMT}}{=} \langle \boldsymbol{a}|\boldsymbol{a}'\rangle \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{b}]f[\boldsymbol{c}\rangle) \\
&= \mathrm{nfSmt}(\langle \boldsymbol{a}|\boldsymbol{a}'\rangle) \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{b}]f[\boldsymbol{c}\rangle) \\
&= \mathrm{nfSmt}(\langle \boldsymbol{a}|\boldsymbol{a}'\rangle \; ; \; \langle \boldsymbol{b}]f[\boldsymbol{c}\rangle)
\end{aligned}$$

  For these equalities to hold, we need to show

$$\mathrm{nfSmt}(\langle \boldsymbol{a}]t[\boldsymbol{b}'\rangle) \; ; \; \langle \boldsymbol{b}'|\boldsymbol{b}\rangle \overset{\mathrm{SMT}}{=} \mathrm{nfSmt}(\langle \boldsymbol{a}]t[\boldsymbol{b}\rangle)$$

  which follows by induction on t:

  * If $t = [\boldsymbol{a}'\rangle\gamma\langle\boldsymbol{b}'']$ then we have

$$\begin{aligned}
\mathrm{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\gamma\langle\boldsymbol{b}''][\boldsymbol{b}'\rangle) \; ; \; \langle \boldsymbol{b}'|\boldsymbol{b}\rangle &= \langle \boldsymbol{a}|\boldsymbol{a}'\rangle \; ; \; \underline{\gamma} \; ; \; \langle \boldsymbol{b}''|\boldsymbol{b}'\rangle \; ; \; \langle \boldsymbol{b}'|\boldsymbol{b}\rangle \\
&\overset{\mathrm{SMT}}{=} \langle \boldsymbol{a}|\boldsymbol{a}'\rangle \; ; \; \underline{\gamma} \; ; \; \langle \boldsymbol{b}''|\boldsymbol{b}\rangle \\
&= \mathrm{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}'\rangle\gamma\langle\boldsymbol{b}''][\boldsymbol{b}\rangle)
\end{aligned}$$

  * If $t = id$ or $t = \delta_{ab}$ then we have

$$\mathrm{nfSmt}(\langle a]\delta_{ab}[b\rangle) \; ; \; \langle b|b\rangle \overset{\mathrm{SMT}}{=} \mathrm{nfSmt}(\langle a]\delta_{ab}[b\rangle)$$

    immediately.

  * If $t = f \; ; \; g$ then we have

$$\begin{aligned}
\mathrm{nfSmt}(\langle \boldsymbol{a}]f \; ; \; g[\boldsymbol{b}'\rangle) \; ; \; \langle \boldsymbol{b}'|\boldsymbol{b}\rangle &= \mathrm{nfSmt}(\langle \boldsymbol{a}]f[\boldsymbol{c}\rangle) \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{c}]g[\boldsymbol{b}'\rangle) \; ; \; \langle \boldsymbol{b}'|\boldsymbol{b}\rangle \\
&\overset{\mathrm{SMT}}{=} \mathrm{nfSmt}(\langle \boldsymbol{a}]f[\boldsymbol{c}\rangle) \; ; \; \mathrm{nfSmt}(\langle \boldsymbol{c}]g[\boldsymbol{b}\rangle) \\
&= \mathrm{nfSmt}(\langle \boldsymbol{a}]f \; ; \; g[\boldsymbol{b}\rangle)
\end{aligned}$$

* If $t = f \uplus g$ then we have

$$
\begin{aligned}
&\mathsf{nfSmt}(\langle \boldsymbol{a}] f \uplus g[\boldsymbol{b}'\rangle) \mathbin{;} \langle \boldsymbol{b}' | \boldsymbol{b}\rangle \\
&= \langle \boldsymbol{a} | \boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \mathbin{;} (\mathsf{nfSmt}\,(\langle \boldsymbol{a}_1] f[\boldsymbol{b}_1\rangle) \oplus \mathsf{nfSmt}\,(\langle \boldsymbol{a}_2] g[\boldsymbol{b}_2\rangle)) \mathbin{;} \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2 | \boldsymbol{b}'\rangle \mathbin{;} \langle \boldsymbol{b}' | \boldsymbol{b}\rangle \\
&\overset{\mathsf{SMT}}{=} \langle \boldsymbol{a} | \boldsymbol{a}_1 \mathbin{+\!\!+} \boldsymbol{a}_2\rangle \mathbin{;} (\mathsf{nfSmt}\,(\langle \boldsymbol{a}_1] f[\boldsymbol{b}_1\rangle) \oplus \mathsf{nfSmt}\,(\langle \boldsymbol{a}_2] g[\boldsymbol{b}_2\rangle)) \mathbin{;} \langle \boldsymbol{b}_1 \mathbin{+\!\!+} \boldsymbol{b}_2 | \boldsymbol{b}\rangle \\
&= \mathsf{nfSmt}(\langle \boldsymbol{a}] f \uplus g[\boldsymbol{b}\rangle)
\end{aligned}
$$

* If $t = \pi \cdot f$ then we have

$$
\begin{aligned}
\mathsf{nfSmt}(\langle \boldsymbol{a}] \pi \cdot f[\boldsymbol{b}'\rangle) \mathbin{;} \langle \boldsymbol{b}' | \boldsymbol{b}\rangle &= \mathsf{nfSmt}(\langle \pi^{-1} \cdot \boldsymbol{a}] f[\pi^{-1} \cdot \boldsymbol{b}'\rangle) \mathbin{;} \langle \boldsymbol{b}' | \boldsymbol{b}\rangle \\
&\overset{\mathsf{SMT}}{=} \mathsf{nfSmt}(\langle \pi^{-1} \cdot \boldsymbol{a}] f[\pi^{-1} \cdot \boldsymbol{b}'\rangle) \mathbin{;} \langle \pi^{-1} \cdot \boldsymbol{b}' | \pi^{-1} \cdot \boldsymbol{b}\rangle \\
&\overset{\mathsf{SMT}}{=} \mathsf{nfSmt}(\langle \pi^{-1} \cdot \boldsymbol{a}] f[\pi^{-1} \cdot \boldsymbol{b}\rangle) \\
&= \mathsf{nfSmt}(\langle \boldsymbol{a}] \pi \cdot f[\boldsymbol{b}\rangle)
\end{aligned}
$$

We also need to show

$$
\mathsf{nfSmt}(\langle \boldsymbol{a}][\boldsymbol{a}' | \boldsymbol{b}][\boldsymbol{b}\rangle) \overset{\mathsf{SMT}}{=} \langle \boldsymbol{a} | \boldsymbol{a}'\rangle
$$

which follows by induction on $\boldsymbol{a}'$:

* If $\boldsymbol{a}' = [a]$ the equality is trivially true
* If $\boldsymbol{a}' = a : \boldsymbol{as}$ we have $\boldsymbol{b} = b : \boldsymbol{bs}$

$$
\begin{aligned}
&\mathsf{nfSmt}(\langle \boldsymbol{a}][a : \boldsymbol{as} | \boldsymbol{b}][\boldsymbol{b}\rangle) \\
&= \mathsf{nfSmt}(\langle \boldsymbol{a}] \delta_{ab} \uplus [\boldsymbol{as} | \boldsymbol{bs}][\boldsymbol{b}\rangle) \\
&= \langle \boldsymbol{a} | a : \boldsymbol{as}'\rangle \mathbin{;} (\mathsf{nfSmt}(\langle a] \delta_{ab}[b\rangle) \oplus \mathsf{nfSmt}(\langle \boldsymbol{as}'][\boldsymbol{as} | \boldsymbol{bs}][\boldsymbol{bs}\rangle)) \mathbin{;} \langle b : \boldsymbol{bs} | \boldsymbol{b}\rangle \\
&= \langle \boldsymbol{a} | a : \boldsymbol{as}'\rangle \mathbin{;} (id \oplus \mathsf{nfSmt}(\langle \boldsymbol{as}'][\boldsymbol{as} | \boldsymbol{bs}][\boldsymbol{bs}\rangle)) \\
&\overset{\mathsf{SMT}}{=} \langle \boldsymbol{a} | a : \boldsymbol{as}'\rangle \mathbin{;} (id \oplus \langle \boldsymbol{as}' | \boldsymbol{as}\rangle) \\
&\overset{\mathsf{SMT}}{=} \langle \boldsymbol{a} | a : \boldsymbol{as}'\rangle \mathbin{;} \langle a : \boldsymbol{as}' | a : \boldsymbol{as}\rangle \\
&\overset{\mathsf{SMT}}{=} \langle \boldsymbol{a} | a : \boldsymbol{as}\rangle
\end{aligned}
$$

– $\mathsf{nfSmt}[\mathbf{SMT}] \subseteq \mathcal{T}h(\mathsf{nfSmt}[\mathsf{dia}[E]] \cup \mathbf{SMT})$ Follows immediately.

□

To conclude this section, we give an analogous result to thm. 7.50 below.

Theorem 7.55. *[Completeness of SMTs]*

Given an $\underset{\sim}{\mathsf{NMT}}\ \langle\Sigma, E\rangle$*, which is complete for some* $\mathsf{n}\mathbb{C}$*, s.t.* $\underset{\sim}{\mathsf{ORD}}(\mathsf{n}\mathbb{C}) \cong \mathbb{C}$*, we show that* $\underset{\sim}{\mathsf{Smt}}\ \langle\Sigma_{\mathbb{F}}, E_{\mathbb{F}}\rangle$ *is complete for* $\mathbb{C}$*.*

Proof. From lem. 7.54 we know that

$$\underset{\sim}{\mathsf{ORD}}(\underset{\sim}{\mathsf{nProp}}\ \langle\Sigma, E\rangle) \cong \underset{\sim}{\mathsf{Prop}}(\underset{\sim}{\mathsf{Smt}}\ \langle\Sigma, E\rangle)$$

From completeness of $\langle\Sigma, E\rangle$ for $\mathsf{n}\mathbb{C}$ we know

$$\underset{\sim}{\mathsf{nProp}}\ \langle\Sigma, E\rangle \cong \mathsf{n}\mathbb{C}$$

Putting these together, we obtain

$$\underset{\sim}{\mathsf{Prop}}(\underset{\sim}{\mathsf{Smt}}\ \langle\Sigma, E\rangle) \cong \mathbb{C}$$

□

The theorem above can be useful when trying to prove soundness and completeness of an ordinary $\underset{\sim}{\mathsf{SMT}}$, where the presented $\underset{\sim}{\mathsf{NMT}}$ is easier to prove sound and complete, like in the case of bijections, discussed previously at the end of sec. 7.3.

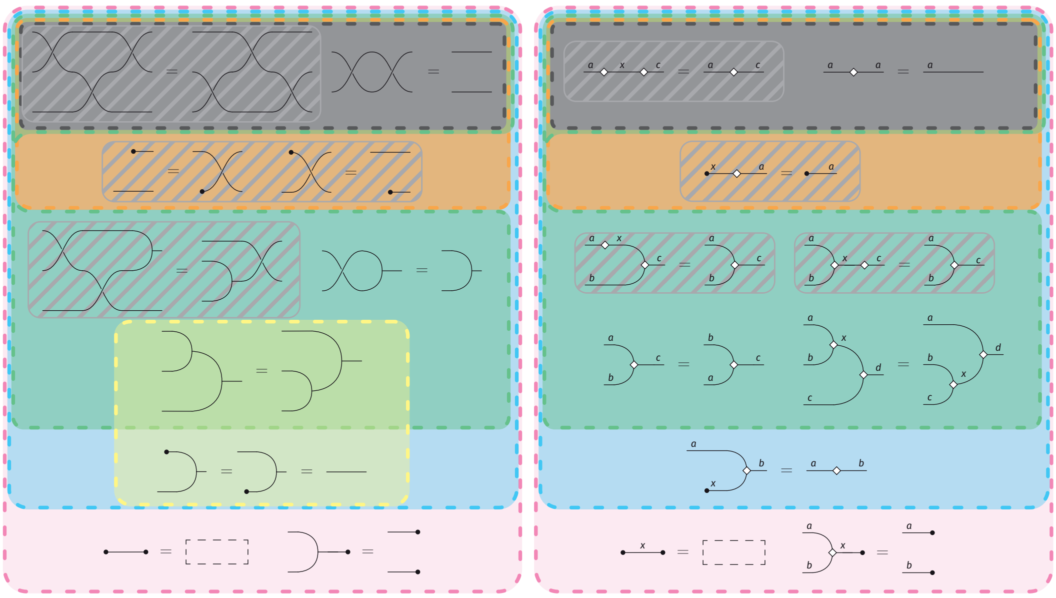
7.7 Related work

Whilst our work is novel in its presentation of nominal string diagrams as monoidal categories internal in $\underset{\sim}{\mathsf{Nom}}$, we are by no means the first to generalise $\underset{\sim}{\mathsf{PROP}}$s to a multi-sorted or nominal settings. Indeed, even (one of) the earliest papers on string diagrams, namely that of Roger Penrose [11], already introduces "nominal" string diagrams where the wires of his pictures are given labels. Amongst later works, the most commonly seen variation to ordinary string diagrams is the notion of colored props [53, 61]. Whilst similar in some aspects to nominal string diagrams, colored string diagrams are somewhat orthogonal to named string diagrams, in that coloured string diagrams usually still have ordered sets of wires, but a nominal variant could be considered with a set of named wires.

Finally, we must mention the work of Blute et al. [62], which is similar in many aspects to our work, especially in the use of the diamond notation $\langle -] - [- \rangle$, which we arrived at independently from the authors, before learning of their work. Another paper in similar spitrit, by Ghica and Lopez [63], introduces a version of nominal string diagrams by explicitly introducing names and binders for ordinary string diagrams.

7.8 Conclusion

The equivalence of nominal and ordinary PROPs (thm. 7.39) has a satisfactory graphical interpretation. Indeed, comparing figs. 7.3, 7.4, truncated and shown side by side below, we see that both share, modulo different labellings of wires mediated by the functors ORD and NOM, the same core of generators and equations. The difference lies only in the equations expressing, on the one hand, that $\oplus$ has natural symmetries and, on the other hand, that generators are a nominal set. In fact, this can be taken as a justification of the importance of naturality, which, informally speaking, compensates for the irrelevant detail induced by ordering names.



There are several directions for future research. First, the notion of an internal monoidal category has been developed because it is easier to prove the basic results in general rather than only in the special case of nominal sets. Nevertheless, it would be interesting to explore whether there are other interesting instances of internal monoidal categories.

Second, internal monoidal categories are a principled way to build monoidal categories with a partial tensor. For example, by working internally in the category of nominal sets with the separated product we can capture in a natural way constraints such as the tensor $f \oplus g$ for two partial maps $f, g : \mathcal{N} \to V$ being defined only if the domains of f and g are disjoint. This reminds us of the work initiated by O'Hearn and Pym on categorical and algebraic models for separation logic and other resource logics, see e.g. [64–66]. It seems promising to investigate how to build categorical models for resource logics based on internal monoidal theories. In one direction, one could extend the work of Curien and Mimram [50] to partial monoidal categories.

Third, there has been substantial progress in exploiting Lack's work on composing PROPs [60] in order to develop novel string diagrammatic calculi for a wide range of applications, see e.g. [51, 67]. It will be interesting to explore how much of this technology can be transferred from PROPs to nominal PROPs.

Fourth, various applications of nominal string diagrams could be of interest. The original motivation for our work was to obtain a convenient calculus for simultaneous substitutions that can be integrated with multi-type display calculi [31] and, in particular, with the multi-type display calculus for first-order logic of Tzimoulis [68]. Another direction for applications comes from the work of Ghica and Lopez [63] on a nominal syntax for string diagrams. In particular, it would be of interest to add various binding operations to nominal PROPs.

La bonne cuisine est la base du véritable bonheur.

Auguste Escoffier

8 Conclusion

As was already mentioned in the introduction, this thesis is split into two sections, corresponding roughly to the two main topics tackled throughout this PhD. In this concluding chapter, we give a chronological overview of how this work developed.

The work on display logics, specifically the calculus toolbox, started in my undergraduate studies as the final year project. When I started my PhD, the original topic was "Efficient methods of proof search for display calculi", where the aim was to explore known efficient proof search tactics for certain logics, such as semantic tableaux or proof search tactics like focusing, and extrapolating these tactics for display calculi. During this initial exploration, it became apparent that the initial version of the calculus toolbox was too fragile for useful exploration of various different display logics. The original toolbox was also insufficient to formalise multi-type display calculi, like the D.EAK [7]. Thus the calculus toolbox 2, described in ch. 3, was born.

Working with our collaborators on a display version of first order logic, our focus shifted to string diagrams, explored in the second part of this thesis, for two main reasons. The first was the excellent introductory course on string diagrams, given by Paweł Sobociński. Inspired by his course and stemmed from a need for a calculus of simultaneous substitutions for our work on a display version of FOL, we started exploring the topic of nominal string diagrams.

Our initial approach to a categorical underpinning for string diagrams featuring labeled wires has been presented in ch. 6. The aim of this work was to give a rigorous account of the partiality of the parallel tensor $\uplus$, inherent in the restriction of only stacking diagrams with disjoint wires/ports. This led us to the notion of partial monoids, which we turned into a categorical notion of partial monoidal categories. The rest of ch. 6 tests these ideas on concrete examples, by adapting ordinary string diagrams to the nominal setting following Lafont [49].

It may seem that ch. 7 supersedes ch. 6, but this is not entirely the case. Whilst ch. 7 does indeed present a refined notion of nominal string diagrams by introducing nominal monoidal categories, partially monoidal categories are in some sense more general than nominal monoidal categories and would make for an interesting topic of further study outside of the scope of nominal string diagrams.

In ch. 7, we simplified the formalism of nominal string diagrams by turning the tensor $\uplus$ back into a total operation, now within the setting of nominal sets. Using internal monoidal categories also helped us simplify the underlying categories of nominal string diagrams; see rem. 7.25 vs def. 6.8 for comparison. This chapter also improves upon the earlier techniques of showing completeness for NMTs, by giving a recipe for translating an ordinary SMT along with it's completeness with respect to some category $\mathbb{C}$, to an NMT which is complete with respect to some nominal version of $\mathbb{C}$. Whilst we have only shown this construction for basic theories, there are many SMTs, which could potentially benefit from this "nominalisation", for example, this recent work by Jacobs and Zanasi [69]; see also [70, 71].

In the last year of the PhD, paralleling the work on nominal string diagrams, we developed yet another new version of the calculus toolbox. Just as the work on nominal string diagrams stemmed out of a need for a calculus of substitutions for the display version of FOL, calculus toolbox 3 was designed to address the difficulties associated with formalising this calculus in a computer. Unlike the previous version of multi-type display calculi, **DFOL** presented unique challenges, which the previous version of the calculus toolbox couldn't effectively deal with.

This thesis represents the last three and a half years of my PhD and looking back, I have fond memories of this time. Having obtained some answers, there are yet more interesting questions which could fill several more PhD's.

Bibilography

[1] G. Gentzen, **Untersuchungen über das logische Schließen I**, *Mathematische Zeitschrift*, vol. 39, pp. 176–210, 1935. DOI: 10.1007/BF01201353

[2] N. Belnap, **Display Logic**, *Journal of Philosophical Logic*, vol. 11, pp. 375–417, 1982. DOI: 10.1007/BF00284976

[3] M. Kracht, **Power and Weakness of the Modal Display Calculus**, in *Applied Logic Series*, Springer Netherlands, 1996, pp. 93–121. DOI: 10.1007/978-94-017-2798-3_7

[4] H. Wansing, **Displaying Modal Logic**. Springer Netherlands, 1998. DOI: 10.1007/978-94-017-1280-4

[5] R. Gore, **Substructural Logics on Display**, *Logic Journal of IGPL*, vol. 6, no. 3, pp. 451–504, May 1998. DOI: 10.1093/jigpal/6.3.451

[6] J. Brotherston, **Bunched Logics Displayed**, *Studia Logica*, vol. 100, no. 6, pp. 1223–1254, Oct. 2012. DOI: 10.1007/s11225-012-9449-0

[7] S. Frittella, G. Greco, A. Kurz, A. Palmigiano, and V. Sikimic, **A proof-theoretic semantic analysis of dynamic epistemic logic**, *Journal of Logic and Computation*, vol. 26, no. 6, pp. 1961–2015, 2016. DOI: 10.1093/logcom/exu063

[8] S. Balco, S. Frittella, G. Greco, A. Kurz, and A. Palmigiano, **Software Tool Support for Modular Reasoning in Modal Logics of Actions** in *Interactive theorem proving*, 2018, pp. 48–67. DOI: 10.1007/978-3-319-94821-8_4

[9] G. Hotz, **Eine Algebraisierung des Syntheseproblems von Schaltkreisen I**, *Elektronische Informationsverarbeitung und Kybernetik*, vol. 1, pp. 185–205, 1965

[10] J. H. Przytycki, **Classical Roots of Knot Theory**, *Chaos, Solitons & Fractals*, vol. 9, nos. 4-5, pp. 531–545, 1998. DOI: 10.1016/s0960-0779(97)00107-0

[11] R. Penrose, **Applications of Negative Dimensional Tensors** in *Combinatorial Mathematics and its Applications*, 1971, pp. 221–244. available at: https://www2.math.uic.edu/~kauffman/Penrose.pdf

[12] A. Joyal and R. Street, **Braided Tensor Categories**, *Advances in Mathematics*, vol. 102, no. 1, pp. 20–78, Nov. 1993. DOI: 10.1006/aima.1993.1055

[13] A. Joyal and R. Street, **The geometry of tensor calculus, I**, *Advances in Mathematics*, vol. 88, no. 1, pp. 55–112, 1991. DOI: 10.1016/0001-8708(91)90003-P

[14] J. R. B. Cockett and R. A. G. Seely, **Proof theory of the cut rule**, in *Categories for the Working Philosopher*, Oxford University Press, 2018. DOI: 10.1093/oso/9780198748991.003.0010

[15] J.-Y. Girard, **Linear logic**, *Theoretical Computer Science*, vol. 50, no. 1, pp. 1–101, 1987. DOI: 10.1016/0304-3975(87)90045-4

[16] P.-A. Melliès, **Functorial Boxes in String Diagrams**, in *Computer science logic*, Springer Berlin Heidelberg, 2006, pp. 1–30. DOI: 10.1007/11874683_1

[17] R. Milner, **Pure bigraphs: Structure and dynamics**, *Information and Computation*, vol. 204, no. 1, pp. 60–122, Jan. 2006. DOI: 10.1016/j.ic.2005.07.003

[18] S. Mason, **Feedback Theory-Some Properties of Signal Flow Graphs**, *Proceedings of the IRE*, vol. 41, no. 9, pp. 1144–1156, Sep. 1953. DOI: 10.1109/jrproc.1953.274449

[19] F. Bonchi and F. Sobociński Paweł and Zanasi, **Full Abstraction for Signal Flow Graphs** in *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2015, pp. 515–526. DOI: 10.1145/2676726.2676993

[20] B. Coecke and A. Kissinger, **Picturing quantum processes: A first course in quantum theory and diagrammatic reasoning**. Cambridge University Press, 2017. DOI: 10.1017/9781316219317

[21] S. Mac Lane, **Categorical algebra**, *Bulletin of the American Mathematical Society*, vol. 71, no. 1, pp. 40–106, Jan. 1965. DOI: 10.1090/S0002-9904-1965-11234-4

[22] P. Selinger, **A Survey of Graphical Languages for Monoidal Categories**, in *New Structures for Physics*, Springer Berlin Heidelberg, 2010, pp. 289–355. DOI: 10.1007/978-3-642-12821-9_4

[23] M. J. Gabbay, **A Theory of Inductive Definitions With α-equivalence**, PhD thesis, University of Cambridge, UK, 2001. available at: http://www.gabbay.org.uk/papers/thesis.pdf

[24] A. M. Pitts, **Nominal sets: Names and symmetry in computer science**. Cambridge University Press, 2013. DOI: 10.1017/CBO9781139084673

[25] B. Lellmann and D. Pattinson, **Correspondence between Modal Hilbert Axioms and Sequent Rules with an Application to S5**, in *Lecture notes in computer science*, Springer Berlin Heidelberg, 2013, pp. 219–233. DOI: 10.1007/978-3-642-40537-2_19

[26] A. Avron, **The Method of Hypersequents in the Proof Theory of Propositional Non-Classical Logics**, in *Logic: From foundations to applications: European logic colloquium*, USA: Clarendon Press, 1996, pp. 1–32

[27] A. Tiu, **A System of Interaction and Structure II: The Need for Deep Inference**, *Logical Methods in Computer Science*, vol. 2, no. 2, Apr. 2006. DOI: 10.2168/lmcs-2(2:4)2006

[28] R. Ramanayake, **Embedding the hypersequent calculus in the display calculus**, *Journal of Logic and Computation*, vol. 25, no. 3, pp. 921–942, Oct. 2014. DOI: 10.1093/logcom/exu061

[29] C. Rauszer, **A Formalization of the Propositional Calculus of H-B Logic**, *Studia Logica*, vol. 33, no. 1, pp. 23–34, Mar. 1974. DOI: 10.1007/bf02120864

[30] H. Wansing, **Constructive negation, implication, and co-implication**, *Journal of Applied Non-Classical Logics*, vol. 18, nos. 2-3, pp. 341–364, Jan. 2008. DOI: 10.3166/jancl.18.341-364

[31] S. Frittella, G. Greco, A. Kurz, A. Palmigiano, and V. Sikimic, **Multi-type display calculus for dynamic epistemic logic**, *Journal of Logic and Computation*, vol. 26, no. 6, pp. 2017–2065, 2016. DOI: 10.1093/logcom/exu068

[32] J. Plaza, **Logics of public communications**, *Synthese*, vol. 158, no. 2, pp. 165–179, Sep. 2007. DOI: 10.1007/s11229-007-9168-7

[33] R. Fagin, **Reasoning About Knowledge**. The MIT Press, 1995

[34] M. Sadrzadeh, A. Palmigiano, and M. Ma, **Algebraic semantics and model completeness for intuitionistic public announcement logic** in *Logic, rationality, and interaction*, 2011, pp. 394–395

[35] A. Tzimoulis, **Algebraic and Proof-Theoretic Foundations of the Logics for Social Behaviour**, PhD thesis, Delft University of Technology, 2018. DOI: 10.4233/uuid:e67e7724-b378-4ca3-ad4e-c40df245af5e

[36] S. T. Kuhn, **Quantifiers as modal operators**, *Studia Logica*, vol. 39, no. 2, pp. 145–158, Jun. 1980. DOI: 10.1007/BF00370318

[37] J. van Benthem, **Modal Foundations for Predicate Logic**, *Logic Journal of IGPL*, vol. 5, no. 2, pp. 259–286, 1997. DOI: 10.1093/jigpal/5.2.259

[38] R. Montague, **Logical necessity, physical necessity, ethics, and quantifiers**, *Inquiry: An Interdisciplinary Journal of Philosophy*, vol. 3, nos. 1-4, pp. 259–269, 1960. DOI: 10.1080/00201746008601312

[39] F. W. Lawvere, **Adjointness in Foundations**, *Dialectica*, vol. 23, pp. 281–296, May 1969. DOI: 10.1111/j.1746-8361.1969.tb01194.x

[40] F. W. Lawvere, **Equality in hyperdoctrines and comprehension schema as an adjoint functor** in *Proceedings of the AMS Symposium on Pure Mathematics XVII*, 1970, pp. 1–14. available at: https://ncatlab.org/nlab/files/LawvereComprehension.pdf

[41] A. Löh, C. McBride, and W. Swierstra, **A Tutorial Implementation of a Dependently Typed Lambda Calculus**, *Fundamenta Informaticae*, vol. 102, no. 2, pp. 177–207, Apr. 2010. available at: https://www.andres-loeh.de/LambdaPi/

[42] P. van der Walt and W. Swierstra, **Engineering Proof by Reflection in Agda** in *Implementation and Application of Functional Languages*, 2013, pp. 157–173. DOI: 10.1007/978-3-642-41582-1_10

[43] M. J. Gabbay and A. M. Pitts, **A New Approach to Abstract Syntax with Variable Binding**, *Formal Aspects of Computing*, vol. 13, no. 3, pp. 341–363, Jul. 2002. DOI: 10.1007/s001650200016

[44] P. Selinger, **A Survey of Graphical Languages for Monoidal Categories**, in *New structures for physics*, Springer Berlin Heidelberg, 2011, pp. 289–355. DOI: 10.1007/978-3-642-12821-9_4

[45] S. Balco and A. Kurz, **Nominal String Diagrams** in *8th Conference on Algebra and Coalgebra in Computer Science (CALCO 2019)*, 2019, vol. 139, pp. 18:1–18:20. DOI: 10.4230/LIPIcs.CALCO.2019.18

[46] H. Brandt, **Über eine Verallgemeinerung des Gruppenbegriffes**, *Mathematische Annalen*, vol. 96, no. 1, pp. 360–366, Dec. 1927. DOI: 10.1007/BF01209171

[47] S. Mac Lane, **Categories for the Working Mathematician**. Springer New York, 1978. DOI: 10.1007/978-1-4757-4721-8

[48] V. Pratt, **Modeling concurrency with partial orders**, *International Journal of Parallel Programming*, vol. 15, no. 1, pp. 33–71, Feb. 1986. DOI: 10.1007/BF01379149

[49] Y. Lafont, **Towards an algebraic theory of Boolean circuits**, *Journal of Pure and Applied Algebra*, vol. 184, pp. 257–310, 2003. DOI: 10.1016/S0022-4049(03)00069-0

[50] P.-L. Curien and S. Mimram, **Coherent Presentations of Monoidal Categories**, *Logical Methods in Computer Science*, vol. 13, no. 3, pp. 1–38, Sep. 2017. DOI: 10.23638/LMCS-13(3:31)2017

[51] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi, **Rewriting modulo symmetric monoidal structure** in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, 2016, pp. 710–719. DOI: 10.1145/2933575.2935316

[52] F. Bonchi, F. Gadducci, A. Kissinger, P. Sobociński, and F. Zanasi, **Confluence of Graph Rewriting with Interfaces**, in *Programming languages and systems*, Springer Berlin Heidelberg, 2017, pp. 141–169. DOI: 10.1007/978-3-662-54434-1_6

[53] F. Zanasi, **Rewriting in Free Hypegraph Categories** in *Proceedings Third Workshop on Graphs as Models, GaM@ETAPS*, 2017, pp. 16–30. DOI: 10.4204/EPTCS.263.2

[54] K. Bar, A. Kissinger, and J. Vicary, **Globular: An online proof assistant for higher-dimensional rewriting**, *Logical Methods in Computer Science ; Volume 14*, pp. Issue 1 ; 1860–5974, 2018. DOI: 10.23638/LMCS-14(1:8)2018

[55] A. Kissinger and V. Zamdzhiev, **Quantomatic: A proof assistant for diagrammatic reasoning**, in *Automated deduction - CADE-25*, Springer International Publishing, 2015, pp. 326–336. DOI: 10.1007/978-3-319-21401-6_22

[56] P. Sobocinski, P. W. Wilson, and F. Zanasi, **CARTOGRAPHER: A Tool for String Diagrammatic Reasoning (Tool Paper)** in *8th conference on algebra and coalgebra in computer science (calco 2019)*, 2019, vol. 139, pp. 20:1–20:7. DOI: 10.4230/LIPIcs.CALCO.2019.20

[57] F. Zanasi, **Interacting Hopf Algebras- the Theory of Linear Systems**, PhD Thesis, Ecole Normale Supérieure de Lyon - ENS LYON, 2015. available at: https://tel.archives-ouvertes.fr/tel-01218015

[58] B. Jacobs, **Categorical Logic and Type Theory**. Elsevier Science, 1999

[59] T. Streicher, **Fibred Categories à la Jean Bénabou**. 1999. arXiv: 1801.02927

[60] S. Lack, **Composing PROPs**, *Theory and Applications of Categories*, vol. 13, pp. 147–163, 2004. available at: http://www.tac.mta.ca/tac/volumes/13/9/13-09abs.html

[61] P. Hackney and M. Robertson, **On the Category of Props**, *Applied Categorical Structures*, vol. 23, no. 4, pp. 543–573, Mar. 2014. DOI: 10.1007/s10485-014-9369-4

[62] R. Blute, J. Cockett, R. Seely, and T. Trimble, **Natural deduction and coherence for weakly distributive categories**, *Journal of Pure and Applied Algebra*, vol. 113, no. 3, pp. 229–296, Dec. 1996. DOI: 10.1016/0022-4049(95)00159-x

[63] D. R. Ghica and A. Lopez, **A Structural and Nominal Syntax for Diagrams** in *Proceedings 14th International Conference on Quantum Physics and Logic, QPL 2017*, 2017, pp. 71–83. DOI: 10.4204/EPTCS.266.4

[64] P. W. O'Hearn and D. J. Pym, **The logic of bunched implications**, *The Bulletin of Symbolic Logic*, vol. 5, no. 2, pp. 215–244, 1999. DOI: 10.2307/421090

[65] D. Galmiche, D. Méry, and D. Pym, **The semantics of bi and resource tableaux**, *Mathematical. Structures in Comp. Sci.*, vol. 15, no. 6, pp. 1033–1088, Dec. 2005. DOI: 10.1017/S0960129505004858

[66] B. Dongol, V. B. F. Gomes, and G. Struth, **A Program Construction and Verification Tool for Separation Logic** in *Mathematics of Program Construction - 12th International Conference, Königswinter, Germany*, 2015, pp. 137–158. DOI: 10.1007/978-3-319-19797-5_7

[67] F. Bonchi, P. Sobociński, and F. Zanasi, **The Calculus of Signal Flow Diagrams I: Linear relations on streams**, *Information and Computation*, vol. 252, no. C, pp. 2–29, Feb. 2017. DOI: 10.1016/j.ic.2016.03.002

[68] A. Tzimoulis, **Algebraic and Proof-Theoretic Foundations of the Logics for Social Behaviour**, PhD Thesis, Delft University of Technology, 2018. DOI: 10.4233/uuid:e67e7724-b378-4ca3-ad4e-c40df245af5e

[69] B. Jacobs and F. Zanasi, **The Logical Essentials of Bayesian Reasoning**, 2018. arXiv: 1804.01193

[70] F. Zanasi, **The Algebra of Partial Equivalence Relations**, *Electronic Notes in Theoretical Computer Science*, vol. 325, pp. 313–333, Oct. 2016. DOI: 10.1016/j.entcs.2016.09.046

[71] F. Bonchi, P. Sobociński, and F. Zanasi, **Interacting Hopf algebras**, *Journal of Pure and Applied Algebra*, vol. 221, no. 1, pp. 144–184, Jan. 2017. DOI: 10.1016/j.jpaa.2016.06.002

A

Sequent calculus in **t3**

This is a sample *t3* file defining a small subset of the propositional fragment of the sequent calculus. This file can be found at: https://github.com/goodlyrottenapple/toolbox3/blob/master/src/test_progs/Sequent.t3.

```
infix 2 ⊢ , ++, ≡, ::
infixr 3 → , ;
infixl 4 ∧, ∨

language LaTeX

smt-data F : Type where
    At  : (atom : Name) → F
  | (∧) : (andL : F) → (andR : F) → F
  | (∨) : (orL  : F) → (orR  : F) → F
  | (→) : (impL : F) → (impR : F) → F
  |  ¬  : (neg  : F) → F
end

translation F to LaTeX where
    At a  → "#{a}"
  | a ∧ b → "#{a} \wedge #{b}"
  | a ∨ b → "#{a} \vee #{b}"
  | a → b → "#{a} \to #{b}"
  | ¬ a   → "\neg #{a}"
end

smt-data List : Type → Type where
     ∅  : {a : Type} → List a
  | (;) : {a : Type} → (hd : a) → (tl : List a) → List a
end

translation List to LaTeX where
    ∅ → ""
  | x ; xs → "\cons{#{x}}{#{xs}}"
end
```

```
smt-builtin (≡) [ '= ] : {a : Type} → a → a → Prop end

smt-builtin (::) [ as ] : {a : Type} → a → Type → a end

smt-def (++) : (x : List F) → (y : List F) → List F where
  (ite ( x ≡ (∅ :: (List F)) )
    y
    ( (hd x) ; ((tl x) ++ y) )
  )
end

data (⊢) : List F → List F → Type where
    Id : {a : Name} → (At a) ; ∅ ⊢ (At a) ; ∅
  | AndL1 : {Γ : List F} → {Δ : List F} → {A : F} → {B : F} →
          A ; Γ ⊢ Δ →
      ---------------
      (A ∧ B) ; Γ ⊢ Δ
  | AndL2 : {Γ : List F} → {Δ : List F} → {A : F} → {B : F} →
          B ; Γ ⊢ Δ →
      ---------------
      (A ∧ B) ; Γ ⊢ Δ
  | OrR1 : {Γ : List F} → {Δ : List F} → {A : F} → {B : F} →
          Γ ⊢ A ; Δ →
      ---------------
      Γ ⊢ (A ∨ B) ; Δ
  | OrR2 : {Γ : List F} → {Δ : List F} → {A : F} → {B : F} →
          Γ ⊢ B ; Δ →
      ---------------
      Γ ⊢ (A ∨ B) ; Δ
  | OrL : {Γ₁ : List F} → {Γ₂ : List F} →
          {Δ₁ : List F} → {Δ₂ : List F} →
          {Γ : List F} → [Γ ≡ (Γ₁ ++ Γ₂)] →
          {Δ : List F} → [Δ ≡ (Δ₁ ++ Δ₂)] →
          {A : F} → {B : F} →
      (A ; Γ₁) ⊢ Δ₁ → (B ; Γ₂) ⊢ Δ₂ →
      ------------------------------
              (A ∨ B) ; Γ ⊢ Δ
  | CR : {Γ : List F} → {Δ : List F} → {A : F} →
      Γ ⊢ A ; A ; Δ →
      -------------
        Γ ⊢ A ; Δ
end

translation (⊢) to LaTeX where
    Id       : x ⊢ y →
      "\AXC{}\RightLabel{$Id$}\n\UIC{$#{x} \vdash #{y}$}"
  | AndL1 p : x ⊢ y →
      "#{p}\RightLabel{$\wedge_{L1}$}\n\UIC{$#{x} \vdash #{y}$}"
  | AndL2 p : x ⊢ y →
      "#{p}\RightLabel{$\wedge_{L2}$}\n\UIC{$#{x} \vdash #{y}$}"
  | OrR1 p  : x ⊢ y →
      "#{p}\RightLabel{$\vee_{R1}$}\n\UIC{$#{x} \vdash #{y}$}"
  | OrR2 p  : x ⊢ y →
      "#{p}\RightLabel{$\vee_{R2}$}\n\UIC{$#{x} \vdash #{y}$}"
  | OrL p q : x ⊢ y →
      "#{p}\n\n#{q}\RightLabel{$\vee_L$}\n\BIC{$#{x} \vdash #{y}$}"
  | CR p     : x ⊢ y →
      "#{p}\RightLabel{$C_R$}\n\UIC{$#{x} \vdash #{y}$}"
end
```

```
def pt : At 'a ∨ At 'b ; ∅ ⊢ At 'a ∨ At 'b ; ∅ where
  CR (OrL
        {∅} {∅}
        {At 'a ∨ At 'b ; ∅} {At 'a ∨ At 'b ; ∅}
        (OrR1 Id) (OrR2 Id))
end

translate pt to LaTeX end
```

B

Internal categories

For further details on this topic, see e.g. Borceux, Handbook of Categorical Algebra I (Chapter 8) or the nLab entry on internal categories.

Definition B.1. [Internal category]

In a category with finite limits an *internal category* is a diagram

$$\begin{tikzcd} A_3 \arrow[r, shift left=4.5, "\mathbf{right}"] \arrow[r, shift left=1.5, "\mathbf{compr}"] \arrow[r, shift right=1.5, "\mathbf{compl}"] \arrow[r, shift right=4.5, "\mathbf{left}"] & A_2 \arrow[r, shift left=3, "\pi_2"] \arrow[r, "\mathbf{comp}"] \arrow[r, shift right=3, "\pi_1"] & A_1 \arrow[r, shift left=3, "\mathbf{dom}"] \arrow[r, shift right=3, "\mathbf{cod}"] & A_0 \arrow[l, "i"] \end{tikzcd} \tag{B.1}$$

such that the following equations hold

1) the diagram $\begin{tikzcd} A_2 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A_1 \arrow[d, "\mathbf{dom}"] \\ A_1 \arrow[r, "\mathbf{cod}"] & A_0 \end{tikzcd}$ is a pullback,
2) $\mathbf{dom} \circ \mathbf{comp} = \mathbf{dom} \circ \pi_1$ and $\mathbf{cod} \circ \mathbf{comp} = \mathbf{cod} \circ \pi_2$
3) $\mathbf{dom} \circ i = \mathrm{id}_{A_0} = \mathbf{cod} \circ i$
4) $\mathbf{comp} \circ \langle i \circ \mathbf{dom}, \mathrm{id}_{A_1} \rangle = \mathrm{id}_{A_1} = \mathbf{comp} \circ \langle \mathrm{id}_{A_1}, i \circ \mathbf{cod} \rangle$
5) $\mathbf{comp} \circ \mathbf{compl} = \mathbf{comp} \circ \mathbf{compr}$

and where

- $\langle i \circ \mathbf{dom}, \mathrm{id}_{A_1} \rangle : A_1 \to A_2$ and $\langle \mathrm{id}_{A_1}, i \circ \mathbf{cod} \rangle : A_1 \to A_2$ are the arrows into the pullback A_2 pairing $i \circ \mathbf{dom}, \mathrm{id}_{A_1} : A_1 \to A_1$ and $\mathrm{id}_{A_1}, i \circ \mathbf{cod} : A_1 \to A_1$, respectively.

- the "triple of arrows"-object A_3 is the pullback

$$\begin{tikzcd} A_3 \arrow[r, "\mathbf{right}"] \arrow[d, "\mathbf{left}"'] & A_2 \arrow[d, "\pi_1"] \\ A_2 \arrow[r, "\pi_2"] & A_1 \end{tikzcd}$$

  where, intuitively, **left** "projects out the left two arrows" and **right** "projects out the right two arrows"

- **compl** is the arrow composing the "left two arrows"

$$\begin{tikzcd} A_3 \arrow[drr, bend left, "\pi_2 \circ \mathbf{right}"] \arrow[dr, dotted, "\mathbf{compl}"] \arrow[ddr, bend right, "\mathbf{comp} \circ \mathbf{left}"'] & & \\ & A_2 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A_1 \arrow[d, "\mathbf{dom}"] \\ & A_1 \arrow[r, "\mathbf{cod}"] & A_0 \end{tikzcd}$$

- **compr** is the arrow composing the "right two arrows"

$$\begin{tikzcd} A_3 \arrow[drr, bend left, "\mathbf{comp} \circ \mathbf{right}"] \arrow[dr, dotted, "\mathbf{compr}"] \arrow[ddr, bend right, "\pi_1 \circ \mathbf{left}"'] & & \\ & A_2 \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A_1 \arrow[d, "\mathbf{dom}"] \\ & A_1 \arrow[r, "\mathbf{cod}"] & A_0 \end{tikzcd}$$

Remark B.2. Equations 1) and 2) define A_2 as the 'object of composable pairs of arrows' while 3) and 4) express that the 'object of arrows' A_1 has identities and 5) formalises associativity of composition. Since A_2 and A_3 are pullbacks, the structure is defined completely already by $(A_0, A_1, \mathbf{dom}, \mathbf{cod}, i, \mathbf{comp})$, but including A_3 as well as $\mathbf{compr}, \mathbf{compl}, \mathbf{right}, \mathbf{left}, \pi_2, \pi_1$ helps writing out the equations.

Definition B.3. A morphism $f : A \to B$ between internal categories, an *internal functor*, is a pair (f_0, f_1) of arrows such that the six squares (one for each of π_2, **comp**, π_1, **dom**, **cod**, i)

$$\begin{tikzcd} A_2 \arrow[r, shift left=2, "\pi_2"] \arrow[r, "\mathbf{comp}" description] \arrow[r, shift right=2, "\pi_1"'] \arrow[d, "f_2"'] & A_1 \arrow[r, shift left=2, "\mathbf{dom}"] \arrow[r, shift right=2, "\mathbf{cod}"'] \arrow[d, "f_1"'] & A_0 \arrow[l, "i" description] \arrow[d, "f_0"'] \\ B_2 \arrow[r, shift left=2, "\pi_2"] \arrow[r, "\mathbf{comp}" description] \arrow[r, shift right=2, "\pi_1"'] & B_1 \arrow[r, shift left=2, "\mathbf{dom}"] \arrow[r, shift right=2, "\mathbf{cod}"'] & B_0 \arrow[l, "i" description] \end{tikzcd} \tag{B.2}$$

commute.

Remark B.4.

- Because B_2 is a pullback f_2 is uniquely determined by f_1. In more detail, if $\Gamma \to B_2$ is any arrow then, because B_2 is a pullback, it can be written as a pair

$$\langle l, r \rangle : \Gamma \to B_2 \tag{B.3}$$

  of arrows $l, r : \Gamma \to B_1$ and f_2 is determined by f_1 via

$$f_2 \circ \langle l, r \rangle = \langle f_1 \circ l, f_1 \circ r \rangle \tag{B.4}$$

- Even if f_2 is not needed as part of the structure in the above definition, including f_2 makes it easier to state that f_1 preserves composition.
- Similarly, B_3 is a pullback, and there is a unique arrow f_3 such that (f_0, f_1, f_2, f_3) together make further 4 squares commute, one for each of **right, compr, compl, left**, see (B.1). We may include f_3 in the structure whenever convenient.

Definition B.5. A natural transformation $\alpha : f \to g$ between internal functors $f, g : A \to B$, an *internal natural transformation*, is an arrow $\alpha : A_0 \to B_1$ such that, recalling (B.3),

$$\mathbf{dom} \circ \alpha = f_0 \qquad \mathbf{cod} \circ \alpha = g_0 \qquad \mathbf{comp} \circ \langle f_1, \alpha \circ \mathbf{cod} \rangle = \mathbf{comp} \circ \langle \alpha \circ \mathbf{dom}, g_1 \rangle$$

Remark B.6. Internal categories with functors and natural transformations form a 2-category. We denote by $\mathit{Cat}(\mathcal{V})$ the category or 2-category of categories internal in $\mathcal{V}$. The forgetful functor $\mathit{Cat}(\mathcal{V}) \to \mathcal{C}$ mapping an internal category A to its object of objects A_0 has both left and right adjoints and, therefore, preserves limits and colimits. Moreover, a limit of internal categories is computed component-wise as $(\lim D)_j = \lim(D_j)$ for $j = 0, 1, 2$.

Remark B.7. A monoidal category can be thought of both as a monoid in the category of categories and as a category internal in the category of monoids. To understand this in more detail, note that both cases give rise to the diagram

$$\begin{tikzcd}
A_2 \times A_2 \arrow[r, "\mathbf{comp}\times\mathbf{comp}"] \arrow[d, "m_2"'] & A_1 \times A_1 \arrow[r, shift left, "\mathbf{dom}\times\mathbf{dom}"] \arrow[r, shift right, "\mathbf{cod}\times\mathbf{cod}"'] \arrow[d, "m_1"'] & A_0 \times A_0 \arrow[d, "m_0"'] \\
A_2 \arrow[r, "\mathbf{comp}"] & A_1 \arrow[r, shift left, "\mathbf{dom}"] \arrow[r, shift right, "\mathbf{cod}"'] & A_0
\end{tikzcd}$$

where

- in the case of a monoid A in the category of internal categories, $m = (m_0, m_1, m_2)$ is an internal functor $A \times A \to A$ and, using that products of internal categories are computed component-wise, we have $\mathbf{comp} \circ m_2 = m_1 \circ (\mathbf{comp} \times \mathbf{comp})$, which gives us the interchange law

$$(f; g) \cdot (f'; g') = (f \cdot f'); (g \cdot g')$$

by using (B.4) with m for f and writing $;$ for **comp** and $\cdot$ for m_1;

- in the case of a category internal in monoids we have monoids A_0, A_1, A_2 and monoid homomorphisms i, **dom**, **cod**, **comp** which, if spelled out, leads to the same commuting diagrams as the previous item.