Université Savoie Mont Blanc

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE
Spécialité : **Mathématiques-Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Clovis Eberhart

Thèse dirigée par **Tom Hirschowitz**

préparée au sein du **Laboratoire de mathématiques**
et de l'**Ecole Doctorale de Mathématiques, Sciences et Technologies de l'Information, Informatique**

Catégories et diagrammes de cordes pour les jeux concurrents

Categories and String Diagrams for Concurrent Game Semantics

Thèse soutenue publiquement le **22 juin 2018**,
devant le jury composé de :

Mr. Martin Hyland
Professor, Department of Pure Mathematics and Mathematical Statistics, University of Cambridge, Président

M. Samuel Mimram
Maître de conférences, LIX, École polytechnique, Rapporteur

M. Pierre-Louis Curien
Directeur de recherche CNRS, IRIF, Université Paris Diderot, Examinateur

M. Vincent Danos
Directeur de recherche CNRS, Département d'informatique, ENS, Examinateur

M. Paul-André Melliès
Chargé de recherche CNRS, IRIF, Université Paris Diderot, Examinateur

M. Tom Hirschowitz
Chargé de recherche CNRS, LAMA, Université Savoie Mont Blanc, Directeur de thèse

Thanks

I would like to extend my thanks to all the people I have shared these few years with. To Tom Hirschowitz, my PhD supervisor, whose vast knowledge of category theory and abstract way of thinking never ceased to amaze me, and who has been more of a colleague with whom I had a fruitful collaboration than someone who simply supervised my work. To Krzysztof Worytkiewicz, who was my official supervisor for the first two years of my PhD, even though we worked on different topics. To the members of the LIMD team, past or present, especially those with whom I have interacted the most : Pierre for all the fascinating math problems and anecdotes about computer science, Jaco for his spirit, humour, and common sense, Christophe for his enthusiasm and ability to always randomly generate conversation topics, and Xavier for his common interests in food and a certain Québecois humourist. To the PhD students at LAMA, either past or present : Rodolphe, Pierre, Florian, Lars, Marion, Boulos, Lama, Rémy, Charlotte, Suelen, and all those I have forgotten to mention. More generally, I would like to thank the whole LAMA laboratory for the wonderful years I have spent there. I would like to (unironically) thank the administration at MSTII (and more generally any person who had to deal with me from an administrative point of view) for their patience. Finally, I would like to thank the reporters – Marin Hyland and Samuel Mimram – and examiners – Pierre-Louis Curien, Vincent Danos, and Paul-André Melliès – for reading this manuscript, making enlightening comments about it, and giving me some pointers for future research directions.

Contents

Chapter 1

Introduction

In recent years, there has been increasingly more focus on concurrent programming, following the increase in the average number of cores in a processor and the rise of distributed computing. However, many programs are still unable to use several cores simultaneously, as concurrent computing is much less intuitive than classical computing. One way to make concurrent computing simpler could be to design languages specifically for concurrency, which requires understanding the basic notions behind it. For example, an appealing aspect of functional languages (admittedly not to the average programmer) such as OCaml or Haskell is that they are built on well-understood theories, and these languages can thus be used to test the effectiveness of functional programming techniques on real problems, rather than academical ones. Once these techniques have proved useful, functional programming paradigms can then be added to "mainstream" programming languages such as Python or C++, where they can be used to solve some problems more easily. This work is a contribution to concurrent game semantics, a research area that uses game semantics to understand concurrency better.

1.1 Semantics of Programming Languages

Semantics of programming languages (or simply *semantics* for short) is a field of computer science whose goal is to assign *mathematical meaning* to programs. Indeed, a *term* of a language is just a sequence of symbols, which in itself carries no meaning, and whose meaning is only understood in the context of a particular language. The idea is thus to build mathematical models of programming languages to prove properties of programs.

There are several reasons why one would want to give a mathematical meaning to programs: to prove that programs written in a particular language have a certain property, to prove that a particular program has the intended behaviour, to prove that it terminates within a reasonable amount of time, to prove that two programs have the same behaviour... It is also interesting to study programming languages in light of their link to logics, given by the *Curry-Howard isomorphism* [95]. In its most basic form, it states that types A of a programming language can be seen as propositions $[\![A]\!]$ of a logic and *vice versa.* But the interesting part is: programs of type A in the language correspond to proofs of

$[\![A]\!]$ in the logic. The correspondence is even finer than this, stating that composition of programs correspond to *cuts* in the logic (a cut is a step in a proof that does not prove anything new, for example, introducing a lemma). Finally, it states a dynamic correspondence between proofs and programs, in the sense that *normalisation* (execution) of a program corresponds to *cut-elimination* in the proof (a process that turns a proof into a proof of the same proposition, but without cuts, and which basically corresponds to inlining all the lemmas introduced in the proof). Semantics is thus a way to understand logic better, and *vice versa*.

1.1.1 An (Outdated) Map of Semantics

Let us give a slightly outdated view of semantics (we will then see that the landscape of semantics is more complex today).

Semantics comes in several different flavours, usually depending on the kind of property that one wishes to prove. It has two main branches: *operational semantics*, which describes programs as some kind of machine, and *denotational semantics*, which describes programs as well-known mathematical structures.

Operational Semantics

Operational semantics is probably the representation of programming languages that is closest to the intuitions programmers have about them. It describes programs as sequences of instructions to be executed by a kind of machine. This is indeed very close to what happens inside a computer, though the set of instructions used in operational semantics is meant to abstract away some of the complexity.

There are different forms of operational semantics, but they all reflect the idea described above. Maybe the most widespread one relies on *labelled transition systems* (or LTSs), which are basically graphs whose vertices are the set of all possible program states and whose edges correspond to execution steps of a program that starts in a certain state and ends in another one. Giving reduction rules for formal languages (such as the λ-calculus [11] or the π-calculus [85]) is exactly defining an LTS whose vertices are the terms of the language and edges are possible reductions. For example, here is a very simple LTS for the λ-calculus:

$$\frac{}{(\lambda x.M)N \to M[N/x]} \qquad \frac{M \to M'}{MN \to M'N} \qquad \frac{N \to N'}{MN \to MN'}.$$

Some LTSs are based on *abstract machines* [63]. As the name indicates, they are in some sense even closer to the idea of a machine executing a sequence of instructions, usually executing a program or *term* within a context called a *stack*, and both the term and the stack may be modified by the various instructions the machine executes. For example, the machine may stack the arguments of an application and then unstack them when they are used, which is exactly what this machine for the λ-calculus does:

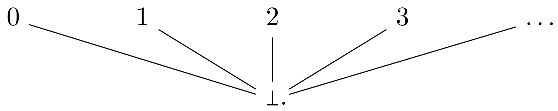
$$MN \star \pi \to M \star N :: \pi \qquad (\lambda x.M) \star N :: \pi \to M[N/x] \star \pi.$$

Here, π is the stack (a sequence of λ-terms), the first rule says that the machine stacks the argument when it encounters an application, and the second rule says that the machine pops an argument and performs the substitution when it encounters an abstraction.

Denotational Semantics

While the interpretation of a program is typically close to its syntax in operational semantics, denotational semantics represents programs as well-known mathematical structures. The idea is to interpret a type A as a space $[\![A]\!]$ of some kind (for example, topological spaces) and functions of type $A \to B$ as morphisms from $[\![A]\!]$ to $[\![B]\!]$ (in the case of topological spaces, morphisms would be continuous functions).

One such semantics is given by *Scott domains* [93]. Basically, a Scott domain is an ordered set that represents "information" about objects of a certain type: $x < y$ means that y holds more information about what it is describing than x. For example, the Scott domain for integers has a bottom element (no information about the integer is known) and an element for each integer (greater than the bottom element, but incomparable to one another) that represents the fact that we know the value of the integer:



A morphism between two Scott domains is basically a monotone function. In terms of programs, this means that, the more information a program has about its input, the more information it may produce about its output. For example, the *denotation* (interpretation) of a program that computes the predecessor but does not terminate on 0 would be the function that maps $\perp$ and 0 to $\perp$, and $n > 0$ to $n - 1$.

A dentoational semantics should respect several properties in order to be considered "good". Let us assume that we are given a relation on terms that expresses whether two terms have the same behaviour (for example, whether they return the same result given the same arguments). The first and most obvious property that a dentoational semantics should verify is *soundness*, i.e., that two programs that have different behaviours must have different denotations. There is not much that can be said about a semantics that does not even verify this property. The second one is *completeness*, i.e., that two programs that have the same behaviour must have equal denotations. When both properties above hold, we say that the model is *fully abstract*. This is an interesting property because studying equality in the model is enough to deduce behavioural equivalence of terms. Another important property is whether the (compact) *definability* result holds in the model, which means that each (compact) element of the model is the interpretation of a term (we also say that the model is *denotationally complete*).

Finally, a crucial property of denotational semantics is *compositionality*, i.e., that the denotation of a program can be deduced from those of its sub-programs. For example, for the λ-calculus, we would want to be able to compute $[\![MN]\!]$ from $[\![M]\!]$ and $[\![N]\!]$. If we see M as a function of type $A \to B$ whose argument is an N of type A, we want to define $[\![MN]\!] = [\![M]\!]([\![N]\!])$, which is indeed compositional. (The actual definition is slightly more involved because we want to interpret typing derivations rather than terms.)

1.1.2 Semantics Today

Today, there is a whole array of models of programming languages, and some of them may be seen both as denotational and operational. A prime example is game semantics: it may be seen as a denotational semantics because it interprets programs as *strategies* on a general notion of game, but strategies actually encode the interaction between the program and its environment, making the model very dynamic, and in many cases (finite) strategies are in bijection with normal forms, making the model very close to syntax, which are some reasons why it is also close to operational semantics.

Nowadays, the meanings of denotational and operational semantics have shifted from their original definitions to take a broader sense. For example, some people do not consider models to be denotational unless they are fully abstract (more precisely, unless they are complete, because models that are not even sound can hardly be called models), but most people consider them to be denotational to some degree. Some models are definitely considered denotational (such as Scott domains) and others definitely not (such as LTSs). Between these two extremes, there is a continuum of models that may be considered more or less denotational, based mainly on two criteria. The first criterion is "how mathematical" the structures used to interpret types and programs are: more common ones (say, topological spaces or vector spaces) will give models that are considered more denotational than models based on less common structures, and *ad hoc* structures (such as LTSs or categories derived from the language's syntax) are not denotational at all. The second criterion is whether the model enjoys "good" features (such as full abstraction) or not: those that do tend to be considered more denotational than those that do not. In both respects, game semantics lies at an intermediate point: strategies are not as common as vector spaces or topological spaces, but they are not *ad hoc* structures either, and while the model is not fully abstract, it is compositional and syntax-independent, and an extensional quotient gives a fully-abstract model.

Similarly, the notion of operational model has also evolved over time. A model used to be considered operational when it was derived from the syntax of a language, such as LTSs. Today, there is another dimension to operational models: a model is considered operational if it is *dynamic*, i.e., the execution of the program can be recovered from its interpretation.

Maybe the distinction between operational and denotational models has become too coarse nowadays, and should be refined into different axes on which each model may be placed: models based on mathematical structures (topological spaces) versus syntactic models (LTSs), static models (functions) versus dynamic models (strategies), or intensional models (equality is based on reduction) versus extensional models (equality is based on observation), etc.

1.2 Game Semantics

We have discussed both operational and denotational semantics (while trying to stay at a rather informal level) and have claimed that game semantics may be seen as both. We here discuss game semantics, which will be at the heart of this work, in a bit more detail.

1.2.1 The Birth of Game Semantics

Game semantics was first born in the realm of logic, in the form of *dialogical logic* [75]. Dialogical logic expresses proofs of a formula as two entities debating whether a formula is true or not: *Proponent*, who tries to prove that the formula is true, and *Opponent*, who tries to prove that it is false. This formal game is the description of a dialogue between two (rational) individuals would have when debating whether a mathematical proposition holds or not: they both defend their case until one of them is convinced they were wrong. A formula is true when Proponent has a *winning strategy* in a certain game played on the formula, which amounts to always managing to convince Opponent that the formula is true, no matter the objections that are raised.

It was then introduced into the world of programming languages under the name *game semantics* by a long series of authors, notably Berry and Curien [13] (under the name of *sequential algorithms*), who were the first to use the idea of interaction in semantics and gave a sequential, denotational semantics of a higher-order language, Blass [14, 15], who exhibited links between game semantics and linear logic [41], Joyal [58], who was the first to build a category of games and strategies, Coquand [23]; who linked game semantics to the dynamics of cut elimination (and thus to evaluation of programs), Abramsky, Jagadeesan, and Malacaria [6], and Hyland and Ong [56] and Nickau [87], who built the most well-known frameworks for game semantics today: AJM and HON games.

The most basic idea comes straight from dialogical logic: types are interpreted as formal games (formulas) and programs as the interactions they may have with the environment (proofs of the formulas). In slightly more detail, types are interpreted as *games* (sometimes called *arenas*) on which notions of *plays* are built. Plays represent all the possible interactions an element of a given type may have with its environment. Programs of a certain type are then interpreted as *strategies* in that game, i.e., sets of plays satisfying some constraints. These plays are the interactions the programs may actually have with the environment: if σ is the interpretation of a program P, then a play p belongs to σ if and only if P may interact with its environment according to p. Here, however, strategies are only used to compute values, and there are no "winning" strategies.

For example, the plays for natural numbers could be sequences of the form $(q\ \mathbb{N})^*$, where q is a *move* in the game representing the environment asking (q stands for "question") for the value of the number, and $\mathbb{N}$ is any natural number, which represents the program answering the value of the number. The strategy corresponding to a counter that increases by 1 each time it is called would consist of all plays of the form $q\ 1\ q\ 2 \ldots q\ n$. For functions of type `int → int`, the set of plays could be of the form $(q_r\ (q_l\ \mathbb{N}_l)^*\ \mathbb{N}_r)^*$, where q_r is a move that represents the environment asking the function for the result (r stands for "right", as in the right-hand side of `int → int`) of its computation, $\mathbb{N}_r$ represents the function returning the value of its result to the environment, q_l represents the function asking for the value of its argument (l stands for "left"), and $\mathbb{N}_l$ represents the environment giving the function the value of its argument. Such a sequence corresponds to the environment asking the function for the value of its result a certain number of times, and each time the function asks for the value of its argument a certain number of times before returning its result. For example, the strategy associated to the successor function would be the set of plays of the form $q_r\ q_l\ n_l^1\ (n^1+1)_r \dots q_r\ q_l\ n_l^k\ (n^k+1)_r$. The exact structure of plays depends on the type of game semantics that we are considering, but this gives a good idea of what plays and strategies look like.

Two other ideas are also present in most game models today. The first one is *innocence*, which is that pure programs (those programs that only use purely functional features) are interpreted as *innocent* strategies. The behaviour of these strategies is based on limited information about what has happened in the play until now. This information basically encodes the part of the interaction between the program and its environment that has led to the current situation. In particular, a function may only rely on the current function call. For example, the strategy for the counter program above is not innocent because it needs to know what it answered last time, and that the only part an innocent strategy would be allowed to rely on is q_r (and the counter is indeed impure). On the other hand, the strategy associated to the successor function is innocent because its answer $(n^k+1)_r$ only depends on n_l^k, and nothing else.

Finally, an important aspect of game semantics is how strategies are composed. Indeed, game models are compositional, and there is in particular a notion of composition of strategies that corresponds to composition of programs. It is defined in two steps called *parallel composition* and *hiding*. Parallel composition lets both strategies interact. To define it, we need to define a notion of "plays" on three games: assume that σ is a strategy on the games A and B and τ a strategy on the games B and C, then we want the parallel composite $\sigma \| \tau$ to be a strategy on the games A, B, and C. The parallel composite then accepts a play on (A,B,C) if and only if its projection to (A,B) is accepted by σ and its projection to (B,C) is accepted by τ. The idea is that σ and τ communicate on the B game. The second step, hiding, consists in erasing the B game to make the composite a strategy only on the games A and C.

As we have already mentioned, game models may be considered denotational because they interpret programs as strategies, which are subsets of general structures. On the other hand, they are operational because strategies are often in bijection with normal forms, and the interpretation of a program is exactly the interactions it may have with the environment, which makes these models close to the dynamics execution.

1.2.2 The Rich World of Game Models

Today, there are many game semantical models based on various ideas that make the game semantical landscape very diverse. What is probably considered the first game semantical model was built by Blass [14, 15]. In this setting, a game is described as the tree of its positions, and plays are branches of that tree. In the case of functions of type $A \to B$, plays may be seen as a branch in A and a branch in B. A strategy is basically a choice of move to play at each node. Composition of strategies is defined by playing on three trees at the same time (then hiding what happens on the middle tree). However, composition is not associative (because of a technical problem with polarity of moves).

The first game semantical models that have become popular and whose variants are still used today are AJM games [6] and HON games [56], which were both invented in the 1990s. AJM games define games as sets of moves together with a condition that tells when a sequence of moves is a valid play. HON games define games as a structured set of moves and plays as structured sequences (the exact structure is beyond the point here). Both models then define strategies as prefix-closed sets of plays and composition of strategies as parallel composition plus hiding, where both operations are defined in similar ways in both models.

Today, there are many variants of HON and AJM games, invented for different purposes. First of all, a number of variations on these games (that impose further conditions on sequences to be valid plays) are described in Harmer's PhD thesis [46]. Some variations are not covered by the reference above, for example [52], in which polarity is reversed. Other variations enrich games with more structure, such as "copycat links" [70], group actions [79, 7, 86], or "coherence" [68]. There are also other models based on the same ideas, but where games can be plugged into other games, such as *polymorphic games* [53], *variable games* [4], *open games* [22], or *context games* [71]. There are other variants that are neither really HON nor AJM, but use the very same ideas, such as the sequoidal category built in [66]. Some models define games as trees (somewhat like Blass games) and strategies as morphisms of games [54, 47].

Furthermore, there are many concurrent game models. This is not surprising in the sense that a fundamental point of concurrency is the interaction between agents, and interaction is precisely at the heart of game semantics. These models may be based on HON games [67, 69, 40, 97] or more exotic structures. One such structure is *event structures* [88], which are more focused on conflict between events than the game models above (thus possibly more suited for concurrency) and which have given many successful game models [92, 20, 21]. Basically, they describe positions as sets of compatible events, moves as adding an event to a position, and strategies as morphisms of games. Melliès has done significant work to give efficient proofs in particular game models with his *asynchronous games* [80, 82, 81, 84], which are also based on event structures.

Another exotic structure is *playgrounds* [50, 29, 30], which spawned the approach used in this thesis. To simplify, they are double categories (basically categories with horizontal and vertical morphisms, and composition is only defined on both classes of morphisms) whose objects are positions of a game, horizontal morphisms are inclusions of positions, and vertical morphisms are plays. They must satisfy a number of properties for this to make sense as a game. From a playground, there is an abstract definition of a category $\mathbb{E}(X)$ of plays starting from the position X. Strategies are then defined as presheaves over $\mathbb{E}(X)$, the idea being that this definition is a generalisation of the traditional notion of prefix-closed sets of plays. The problem is that, to this day, it is still unknown how to compose strategies as parallel composition plus hiding. All known examples of playgrounds are part of a more general framework where plays are defined as *string diagrams* (which are basically formalisations of intuitive drawings used in game semantics). In this framework, moves are defined as basic string diagrams, and plays are defined as pastings of moves. Most of the work described in this thesis is done in this string diagrammatic framework.

There have been very few attempts to understand the world of game models globally. There are however a few notable exceptions. In [18], Bowler defines a general construction of game models and composition of strategies. However, he seems to be more interested in mathematical games than the ones that arise as models of programming languages. In particular, the examples he treats are those of simple games and Conway games, and he does not consider the problem of defining innocent strategies. Finally, in [48], the authors define a general framework of game models in a very close spirit to Chapter 6 of this thesis and composition of strategies abstractly, but the frameworks differs with ours on a number of fundamental points: first, the notion of morphism used there can only model prefix ordering, and second, they do not treat the case of innocent strategies.

1.3 Motivation and Contributions

The main motivation for this work was to understand the landscape of game models better. More precisely, there are many game models, some of them are very similar while others are based on completely different ideas, but there is very little literature that provides insight about the links between all these different games. Our goal was to provide insight about game models, mainly through abstraction: we have tried to find properties that are verified in different game models and abstract them away to create classes of models that verify these properties, and to prove properties for all models of that class. A characteristic feature of our approach, other than abstraction, is our extensive use of advanced categorical tools, which makes constructions and proofs more streamlined and efficient. In this matter, we have benefited from previous work in the same spirit, namely the recent recasting of strategies as presheaves, and of innocence as a sheaf condition, enabled by Melliès's notion of morphisms between plays. A key tool that we introduce for the first time in game semantics is the theory of *exact squares* [45], which proves very efficient in Chapters 5 and 6.

Main Contributions

We here give the main results of each of the contributions that will be discussed in this thesis. Each contribution is given a longer section dedicated to detailing its results and the methods developed to prove them just below.

Fibred Models Our first contribution follows the pattern of abstraction explained above and is discussed in Chapter 4. All known instances of string diagrammatic models (one for CCS [50], one for the π-calculus [29], one for HON games, which we study in Chapter 5, and an unpublished one for the join-calculus [36]) follow the same construction: they first define positions as some kind of graphs, then moves as higher-dimensional arrows, and finally plays as composites of moves. We call such positions and plays *string diagrams*, because they formalise the drawings physicists call so. A crucial property to be able to define a category of plays starting from a fixed position is *fibredness*, which basically states that plays can be canonically restricted to sub-positions. We first give an abstract way to build string diagrammatic models from an operational description of a language that generalises previous constructions. We then give a necessary and sufficient criterion for the fibredness property to hold, and another sufficient criterion that is easier to prove. This contribution is based on [25].

A Bridge Between Models Our second main contribution is a connection between two variations on HON games, as discussed in Chapter 5. The first one [97] is based on the standard notion of *justified sequence*, while the second one follows the string diagrammatic approach developed in our previous contribution. There is an obvious, yet informal link between both models in the sense that they both define innocent strategies as sheaves for a Grothendieck topology induced by embedding views into plays. We show that this relationship can actually be tightened: they are equivalent in the sense that they yield equivalent categories of innocent strategies. We first try to give a slightly informal argument, based on derivation trees in an *ad hoc* sequent calculus describing HON games, and then give a direct, formal proof, without using derivation trees. This contribution is based on [26] (which explains the argument using derivation trees) and its extended version [25] (which gives the direct proof).

A Core of Game Models Our last main contribution, which we discuss in Chapter 6, is the development of a general framework to study game models. The idea is to try to boil down game models to a few basic properties about their plays, and to derive the main constructions of game models from these basic properties. Starting from a structure that represents plays, we show how to define a notion of concurrent strategy and how to compose them by parallel composition plus hiding. We show that, under some mild conditions, composition of strategies is associative and unital. Under further assumptions, we show how to define a category of innocent concurrent strategies. We also show that this framework encompasses many existing game models. This contribution is based on [28].

Contributions Not Discussed in This Thesis

Let us finally mention a few contributions that will either not be detailed in this thesis or only used as examples.

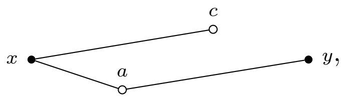
String Diagrams for Concurrent Traces and Unfolding The first such contribution is Chapter 3, which is based on [24], and which we use as an introduction to *string diagrams*, which are extensively used in Chapters 4 and 5. We show how they can be used to model concurrent traces in a simple way, give a few examples this construction applies to, and illustrate it on the example of Petri nets.

A Game Model of the π-calculus In [29] and its extended version [30], we build a model for the π-calculus that is intensionally fully abstract for fair testing equivalence (intensional full abstraction is simply denotational completeness, with the idea that an extensional collapse of the model gives a fully-abstract model, which is necessary in some cases as some abstract models have no recursively enumerable presentation [74]). We first define a notion of play based on string diagrams for the π-calculus (this is actually the example we use in Chapter 4) and show how to interpret terms as strategies. To show that this interpretation is intensionally fully abstract, we appeal to the theory of *playgrounds*, from which we abstractly derive an LTS for our notion of strategy and relate it to that of the π-calculus. Many ideas present in Chapter 4 come from this paper, in particular the use of factorisation systems to prove fibredness.

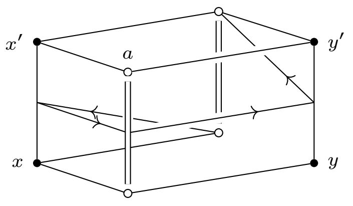
Interpreting Terms as Strategies Abstractly In [27], we study the interpretation of terms into strategies in game models from an abstract point of view. The idea is to define a broad notion of *paraterm* into which views, plays, and terms can all be embedded. We can then abstractly define the interpretation of terms as innocent strategies as a *singular functor*, which abstracts some previous interpretations. We then recover a fundamental result of game semantics, known as *definability* (which states that all finite innocent strategies are isomorphic to the image of a value), under the form of *geometric realisation.*

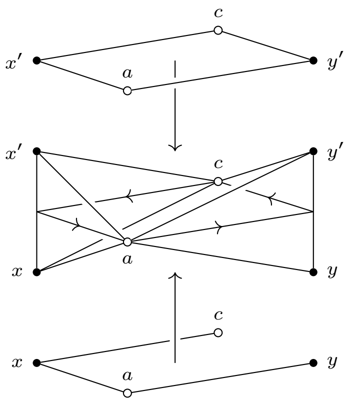
1.3.1 Fibred Models

In Chapter 4, we show how to create string diagrammatic models and show that they are fibred under some conditions. We start from a base category $\mathbb{C}$ describing the operational semantics of a language. This base category comes with a notion of dimension. Objects of lower dimensions are called *players* and *channels* and describe *positions* of the game, which are basically graphs of players and channels. Players are the agents of the game and channels are the means by which they can communicate. For example, in the π-calculus, a position simply represents the topology of communication between agents, as in:



which represents a position with two players x and y who can communicate through the channel a, and x knows a private channel c. Objects of higher dimension describe the dynamics of the game. For example, in the π-calculus, a synchronisation where x sends c on a and y receives it is drawn as on the left below.





Here, the initial position of the synchronisation is drawn at the bottom, the final position at the top, and we can see that, in the final position, the avatar y' of y knows the channel c that x has sent them. For the π-calculus, synchronisation corresponds to an object of higher dimension in the category $\mathbb{C}$. Each object of higher dimension of $\mathbb{C}$ is assigned such a drawing, which we call a *move*. We further call the assignment of all these moves a *signature*. A *play* is a composite (pasting) of such moves. Formally, positions are presheaves over the first two dimensions of $\mathbb{C}$ (this is formal because the drawings represent the *categories of elements* of the objects of $\mathbb{C}$). Moves are cospans $Y \to M \leftarrow X$ of presheaves over $\mathbb{C}$, where X is the initial position, Y is the final position, and M represents the move. For example, the cospan corresponding to the synchronisation in the π-calculus is drawn next to it, with X at the bottom, Y at the top, M in the middle, and the morphisms are inclusions.

From any signature S, we build a pseudo double category $\mathbb{D}_{\mathsf{S}}$, which, to simplify, is a gadget that has a set of *objects* (here, positions), and for all objects X and Y, a set of *horizontal morphisms* $X \to Y$ (here, inclusions of X into Y) and a set of *vertical morphisms* $Y \xrightarrow{\bullet} X$ (here, plays starting from X and ending in Y). It also has, for all perimeters as below, a set of *cells* α (which here represents the fact that u embeds into u' in a certain sense).

$$\begin{tikzcd}
Y \arrow[r, "k"] \arrow[d, "u"', "\bullet" marking] \arrow[rd, phantom, "\overset{\alpha}{\Longrightarrow}"] & Y' \arrow[d, "u'", "\bullet" marking] \\
X \arrow[r, "h"'] & X'
\end{tikzcd}$$

We then want to define a category $\mathbb{E}(X)$ of plays over a fixed position X whose morphisms $u \to u'$ would represent the fact that u' is an extension of u ($\mathbb{E}(X)$ depends on the signature S, but we leave the dependence implicit for readability). The natural definition of morphism from $u{:}\, Y \xrightarrow{\bullet} X$ to $u'{:}\, Y' \xrightarrow{\bullet} X$ is thus a tuple of a vertical morphism $w{:}\, Z \to Y$, a horizontal morphism $h{:}\, Z \to Y'$, and a cell α as in:

$$\begin{tikzcd}
Z \arrow[r, "h"] \arrow[d, "w"', "\bullet" marking] & Y' \arrow[dd, "u'", "\bullet" marking] \\
Y \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] \arrow[d, "u"', "\bullet" marking] & \\
X \arrow[r, equal] & X.
\end{tikzcd}$$

To be able to define a category, we need to be able to compose such morphisms, i.e., to canonically find a dashed part to the solid part of:

$$\begin{tikzcd}
Z'' \arrow[r, dashed] \arrow[d, dashed, "\bullet" marking] \arrow[rd, phantom, "\overset{\beta}{\Longrightarrow}"] & Z' \arrow[r, "h'"] \arrow[d, "w'", "\bullet" marking] & Y'' \arrow[ddd, "u''", "\bullet" marking] \\
Z \arrow[r, "h"] \arrow[d, "w"', "\bullet" marking] & Y' \arrow[dd, "u'", "\bullet" marking] \arrow[r, phantom, "\overset{\alpha'}{\Longrightarrow}"] & \\
Y \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] \arrow[d, "u"', "\bullet" marking] & & \\
X \arrow[r, equal] & X \arrow[r, equal] & X,
\end{tikzcd}$$

so we need to be able to canonically restrict a play w' to a smaller position Z. This is the property we call *fibredness*: for all plays $u: Y \twoheadrightarrow X$ and morphisms $X' \to X$, there must exist a play $u': Y' \twoheadrightarrow X'$ and a cell α as below such that, for all diagrams as the solid part of

$$\begin{tikzcd}
Y'' \arrow[rrd, "k'"] \arrow[rd, dashed, "k''"'] \arrow[dd, "u''"'] & & \\
& Y' \arrow[r, "k"'] \arrow[dd, "u'"'] & Y \arrow[dd, "u"] \\
X'' \arrow[rd, "h''"'] \arrow[rrd, "h'"] & & \\
& X' \arrow[r, "h"'] & X,
\arrow[from=3-1, to=2-2, phantom, "\overset{\alpha''}{\Rightarrow}"]
\arrow[from=2-2, to=3-3, phantom, "\overset{\alpha'}{\Rightarrow}"]
\arrow[from=3-2, to=3-3, phantom, "\overset{\alpha}{\Rightarrow}"]
\end{tikzcd}$$

there is a dashed arrow and corresponding cell (this basically means that u' is really a restriction of u along h and is necessary for composition in $\mathbb{E}(X)$ to be well defined).

To prove it, we appeal to *factorisation systems* [17]. This algebraic tool allows to factor all morphisms of a category as $r \circ l$, where l and r belong to fixed classes $\mathcal{L}$ and $\mathcal{R}$ that are *orthogonal*, which means that they have a certain lifting property. We build a factorisation system whose class $\mathcal{L}$ is generated by the legs $X \to M$ for all cospans $Y \to M \leftarrow X$ in the signature (i.e., for all moves). Remember that a play $u: Y \twoheadrightarrow X$ is a cospan of presheaves $Y \to U \leftarrow X$, so we are actually faced with a situation like the solid part of the left-hand side diagram below, which we complete by factoring $l \circ h$ as $h' \circ l'$ (using the factorisation system) and then taking the pullback of f and h'.

$$\begin{tikzcd}
Y' \arrow[r, dashed, "h''"] \arrow[d, dashed, "f'"'] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d, "f"] \\
U' \arrow[r, dashed, "h'"] & U \\
X' \arrow[u, dashed, "l'"] \arrow[r, "h"] & X \arrow[u, "l"']
\end{tikzcd}
\qquad
\begin{tikzcd}
Y'' \arrow[rrd, bend left, "q''"] \arrow[rd, dashed, "s''"] \arrow[dd, "f''"'] & & \\
& Y' \arrow[r, "h''"] \arrow[dd, "f'"] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[dd, "f"] \\
U'' \arrow[rrd, bend left, "q'"] \arrow[rd, dashed, "s'"] & & \\
& U' \arrow[r, "h'"] & U \\
X'' \arrow[uu, "l''"] \arrow[rd, "s"] \arrow[rrd, bend left, "q"] & & \\
& X' \arrow[uu, "l'"] \arrow[r, "h"] & X \arrow[uu, "l"']
\end{tikzcd}$$

We then get the desired universal property by building s' and s'' in the right-hand side diagram: the former comes from the lifting property of our factorisation system and the latter follows by universal property of pullback.

It then remains to show that $Y' \to U' \leftarrow X'$ is a play, which we prove by induction, under the hypothesis that it is the case for moves. We then give a sufficient criterion on $\mathbb{C}$ for the restriction to be a play.

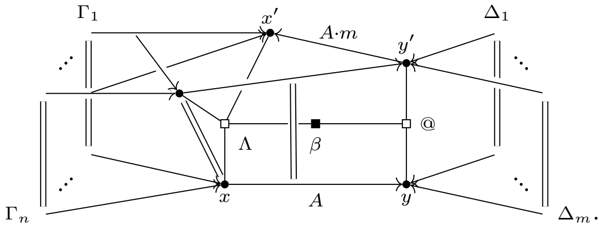
1.3.2 A Bridge Between Models

In Chapter 5, we start by building a string diagrammatic approach to HON games as explained in 1.3.1 and then exhibit links between this model and another one (based on the standard notion of justified sequence) both at the level of plays and at the level of strategies. We start from a sequent calculus that describes arena games, strongly reminiscent of a focalised calculus for intuitionistic logic. From this sequent calculus, we derive a signature S_{HON} that describes HON games. Here, channels are games on which two players play. We draw channels as edges and players as nodes, and the positions typically look like

$$\xrightarrow{A} \overset{x}{\bullet} \xrightarrow{B} \overset{y}{\bullet} \xrightarrow{C},$$

where, in this particular position, x plays as Proponent on B and Opponent on A, and y as Proponent on C and Opponent on B. The dangling arrows represent interaction with the environment. This could typically model the composition of a function of type $A \to B$, modelled by x, and one of type $B \to C$, modelled by y. Players that have only incoming edges are morally the program fragments that are currently computing something, while the ones with an outgoing edge are waiting for another program fragment to call them (on that outgoing edge).

The dynamics of this game is derived from cut elimination in our sequent calculus, and drawn as:



Maybe the simplest way to understand this interaction is from a computational point of view. In the initial position (at the bottom of the drawing), x is a function $\Gamma_1 \to \ldots \to \Gamma_n \to A$ that produces results of type A with access to resources of type $\Gamma_1, \ldots, \Gamma_n$, while y is a program fragment that is currently computing and has access to resources of type $\Delta_1, \ldots, \Delta_m$, and A produced by x. The interaction represents y asking x for its return value. In the final position (at the top of the drawing), the polarities of both players have changed, since x' (the avatar of x after they are called) is now computing and y' is waiting for x' to call it back with its return value (which is why x' has “access” to y': this is just a continuation). Notice that y' still has access to x, in cases it needs to ask it to compute another value later during its execution.

We then derive a pseudo double category $\mathbb{D}_{HON}$ whose objects are positions, horizontal morphisms are inclusions of positions, vertical morphisms are plays, etc, as in 1.3.1. We then show that our signature S_{HON} verifies the conditions for $\mathbb{D}_{HON}$ to be fibred, from which we derive a category of plays $\mathbb{E}(X)$ above any position X. In particular, we get categories $\mathbb{E}(A \vdash B)$ above all positions of the form

$$\xrightarrow{A} \overset{x}{\bullet} \xrightarrow{B}.$$

We also derive subcategories $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ that consist only of *views*, which are particular plays, defined in a slightly *ad hoc* way. Strategies are then standardly defined as presheaves over $\mathbb{E}(A \vdash B)$, and innocent strategies as those presheaves that are in the image of $\prod_{\mathsf{i}} \colon \widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \to \widehat{\mathbb{E}(A \vdash B)}$, where i is the embedding of $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ into $\mathbb{E}(A \vdash B)$, $\widehat{\mathbb{C}}$ denotes the category of presheaves over $\mathbb{C}$, and $\prod$ denotes right Kan extension.

There are standard categories corresponding to these in the variant of HON games based on justified sequences: the category $\mathbb{P}_{A,B}$ of plays on the pair of arenas (A, B), and the category $\mathbb{V}_{A,B}$ of views. Similarly, strategies are defined as presheaves over $\mathbb{P}_{A,B}$, and innocent strategies as the presheaves in the image of $\prod_{\mathsf{i}_{HON}}$ (where i_{HON} is the embedding of $\mathbb{V}_{A,B}$ into $\mathbb{P}_{A,B}$).

Most of the chapter is spent on building a commuting square as below left, where F is a full embedding and $F^{\mathbb{V}}$ is an equivalence of categories.

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r, hook, "\mathsf{i}_{TO}"] \arrow[d, "F^{\mathbb{V}}"'] & \mathbb{P}_{A,B} \arrow[d, "F"] \\
\mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[r, hook, "\mathsf{i}"'] & \mathbb{E}(A \vdash B)
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\mathbb{V}_{A,B}} \arrow[r, hook, "\Pi_{\mathsf{i}_{TO}}"] & \widehat{\mathbb{P}_{A,B}} \\
\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \arrow[u, "\Delta_{F^{\mathbb{V}}}"] \arrow[r, hook, "\Pi_{\mathsf{i}}"'] & \widehat{\mathbb{E}(A \vdash B)} \arrow[u, "\Delta_F"']
\end{tikzcd}$$

In particular, we have that $\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)}$ and $\widehat{\mathbb{V}_{A,B}}$ are equivalent through the restriction functor $\Delta_{F^{\mathbb{V}}}$. But there is more: the fact that $F^{\mathbb{V}}$ is an equivalence and F is fully faithful implies that the square is *exact* [45], which means that the square above right commutes up to isomorphism. This means that the categories of innocent strategies are equivalent in both variants of the model, and that this equivalence is compatible with the *saturation* functors $\prod_{\mathsf{i}}$ and $\prod_{\mathsf{i}_{HON}}$. The differences between both variants is thus mostly a matter of presentation.

The hard part of the chapter is to define the functors F and $F^{\mathbb{V}}$ above. The latter is simply defined by restriction of the former to views, so the most difficult part is to define F. We do this in two ways: we first give a slightly informal argument, and then give a formal proof.

The first way uses derivation trees in an *ad hoc* sequent calculus. We define a category $\mathbb{T}(A \vdash B)$ whose objects are trees of conclusion $(A \vdash B)$ and whose morphisms are inclusions of such trees, and a subcategory $\mathbb{B}(A \vdash B)$ of branches of conclusion $(A \vdash B)$. We decompose the desired square into

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r] \arrow[d, "\mathsf{i}_{TO}"'] & \mathbb{B}(A \vdash B) \arrow[r] \arrow[d] & \mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[d, "\mathsf{i}"] \\
\mathbb{P}_{A,B} \arrow[r] & \mathbb{T}(A \vdash B) \arrow[r] & \mathbb{E}(A \vdash B)
\end{tikzcd}$$

by first showing that $\mathbb{T}(A \vdash B)$ is equivalent to $\mathbb{E}(A \vdash B)$, that this equivalence restricts to an equivalence between $\mathbb{B}(A \vdash B)$ and $\mathbb{E}^{\mathbb{V}}(A \vdash B)$, and then by building a full embedding from $\mathbb{P}_{A,B}$ to $\mathbb{T}(A \vdash B)$ and showing that it restricts to an equivalence between $\mathbb{V}_{A,B}$ and $\mathbb{B}(A \vdash B)$. This is however not entirely satisfactory in the sense that trees are not handled very formally, and a formal definition of $\mathbb{T}(A \vdash B)$ would make the problem as difficult to solve as without using $\mathbb{T}(A \vdash B)$.

We thus then give a formal construction of F and $F^{\mathbb{V}}$ without using $\mathbb{T}(A \vdash B)$. The proofs are much more *ad hoc* than in the rest of this thesis, which is not very surprising in the sense that we are trying to link models that are built using very different methods.

1.3.3 A Core of Game Models

In Chapter 6, we want to build different categories of strategies from a basic description of a game model. We start from some data $\mathbb{P}$ that describes a game model. For all games A, $\mathbb{P}$ gives a category $\mathbb{P}_A$ of plays on the game A. It also gives, for all games A and B, a category $\mathbb{P}_{A,B}$ of plays on the pair (A,B), and similarly for triples and quadruples of games. It also gives insertion functors, for example $\iota_0 : \mathbb{P}_{A,B} \to \mathbb{P}_{A,A,B}$ that typically duplicates what happens on the left-hand side game, and deletion functors, for example $\delta_1 : \mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ that typically erases what happens on the middle game.

We abstractly derive a notion of strategy from $\mathbb{P}$: a strategy on the pair of games (A,B) is a presheaf over $\mathbb{P}_{A,B}$ (we only study strategies on a pair of games because we want to show that they form a category, but we could define strategies on a single game similarly). The idea is that a strategy σ accepts a play p if $\sigma(p) \neq \varnothing$ (more precisely, $\sigma(p)$ is the set of states the strategy can be in after playing p). To show that games and strategies form a category, we then define a composition as parallel composition plus hiding and identities for this composition, known as *copycat strategies*.

The parallel composition $\sigma \| \tau$ of two strategies σ on (A,B) and τ on (B,C) accepts to play an *interaction sequence* (a play on three games) u if and only if σ accepts to play the projection $\delta_2(u)$ of u to (A,B) and τ accepts to play its projection $\delta_0(u)$ to (B,C). The hiding of a strategy σ on $\mathbb{P}_{A,B,C}$ accepts to play p if and only if there is an interaction sequence u that projects to p that σ accepts to play, i.e., it accept the same plays as σ, except we hide what happens on B. Finally, the copycat strategy on the game A is a strategy on $\mathbb{P}_{A,A}$ that "copies" all the moves Opponent plays.

In our framework, both composition and copycats are defined as *polynomial functors*. Given a functor $F : \mathbb{C} \to \mathbb{D}$, the restriction functor $\Delta_F : \widehat{\mathbb{D}} \to \widehat{\mathbb{C}}$ is given by pre-composition by F^{op}. It admits left and right adjoints $\sum_F : \widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$ and $\prod_F : \widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$, called *left* and *right Kan extension*, respectively. The intuition behind $\sum_F$ is that the presheaf $\sum_F(X)$ is non-empty over an element d if there exists an antecedent c of d such that X is non-empty over c. For $\prod_F$, the intuition is that X must be non-empty over all antecedents of d. These functors can thus be seen as $\exists$ and $\forall$ functors. A functor from $\widehat{\mathbb{C}}$ to $\widehat{\mathbb{D}}$ is polynomial if it is a composite of any number of restrictions and left and right Kan extensions.

Composition must be a functor from $\widehat{\mathbb{P}_{A,B}} \times \widehat{\mathbb{P}_{B,C}}$ to $\widehat{\mathbb{P}_{A,C}}$. This is the same thing as a functor from $\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}}$ to $\widehat{\mathbb{P}_{A,C}}$. We define it as the composite

$$\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \xrightarrow{\Delta_{\delta_2+\delta_0}} \widehat{\mathbb{P}_{A,B,C} + \mathbb{P}_{A,B,C}} \xrightarrow{\Pi_\nabla} \widehat{\mathbb{P}_{A,B,C}} \xrightarrow{\Sigma_{\delta_1}} \widehat{\mathbb{P}_{A,C}},$$

where ∇ is the codiagonal functor. This is indeed a polynomial definition, and the idea behind it is exactly that of parallel composition plus hiding, which we can see by computing this functor, using the description of Kan extensions given above. The composite of the first two functors is parallel composition: if it maps the copairing $[\sigma,\tau]$ to θ, then computation shows that θ accepts to play the interaction sequence u if and only if, for all antecedents u' of u (that is $\mathrm{inl}\, u$ and $\mathrm{inr}\, u$), $[\sigma,\tau]$ accepts to play $(\delta_2 + \delta_0)(u')$, i.e., σ accepts to play $\delta_2(u)$ and τ accepts to play $\delta_0(u)$. The last functor $\sum_{\delta_1}$ is hiding: if it maps σ to τ, then τ accepts to play p if and only if there exists an interaction sequence u that projects to p and is accepted by σ.

Copycat strategies are also defined as polynomial functors. The copycat strategy on A may be defined as a functor from 1 to $\widehat{\mathbb{P}_{A,A}}$. Since $1 \cong \widehat{\varnothing}$, we can define it as:

$$\widehat{\varnothing} \xrightarrow{\Pi_!} \widehat{\mathbb{P}_A} \xrightarrow{\Sigma_{\iota_0}} \widehat{\mathbb{P}_{A,A}}.$$

The idea is that $\prod_!$ is the terminal presheaf on $\mathbb{P}_A$, so it accepts all plays in $\mathbb{P}_A$, and $\sum_{\iota_0}(\sigma)$ accepts a play p if and only if p is of the form $\iota_0(p')$ and σ accepts p'. Since ι_0 typically corresponds to copying what happens on A to the other copy of A, the whole composite indeed corresponds to the copycat strategy on A: it accepts to play p if and only if Proponent copies everything Opponent does.

We then set out to show that games and strategies form a category whose composition and identities we have just defined. This means that composition must be associative and that the copycat strategies should be its units. We prove that, under some conditions, composition is associative and copycat strategies are units. The main condition is inspired by the method that is usually used in game models to show that composition of strategies is associative: the *zipping lemma*, which states that, in some cases, given two interaction sequences that project to the same play, there is a unique way to build a *generalised interaction sequence* (a play on four games) that projects to the original interaction sequences.

All this work is done using our notion of concurrent strategy, and we want to get the same results for "traditional" strategies, which we see as functors $\mathbb{P}^{op}_{A,B} \to 2$, where 2 is the ordinal $0 \to 1$ seen as a category. We derive from the fact that games and concurrent strategies form a category that games and traditional strategies also do. We also investigate a number of game models and show that they fit in this framework, and that composition of strategies in these game models corresponds to composition of strategies as defined abstractly in our framework.

Finally, we tackle the question of innocence. We assume that we are given a full subcategory $\mathbb{V}_{A,B} \xrightarrow{\mathrm{i}_{A,B}} \mathbb{P}_{A,B}$ of views to each category $\mathbb{P}_{A,B}$. We then define innocent strategies as those presheaves that are in the image of $\widehat{\mathbb{V}_{A,B}} \xrightarrow{\Pi_{\mathrm{i}_{A,B}}} \widehat{\mathbb{P}_{A,B}}$. The idea of this definition is that a presheaf $\prod_{\mathrm{i}_{A,B}}(\sigma)$ accepts to play p if and only if, for all morphisms $v \to p$ from a view v to p, σ accepts to play v. For this definition to be the right one, $\mathbb{P}_{A,B}$ should contains enough morphisms (in the case of HON games, the notion of morphism is that given by Melliès [80], reused by Levy [73] and Tsukada and Ong [96]). Such a presheaf thus accepts to play p if and only if it accepts to play all its views, which is the idea of innocence. We then set out to show that games and innocent strategies form a subcategory of games and strategies. By adding some properties on the model, we show that this indeed holds.

It may be interesting to see how we prove this kind of results. For example, let us take preservation of innocence, which states that the composite of two innocent strategies is again innocent. We prove this by studying the diagram below. (The reader does not need to understand this diagram.)

$$\begin{tikzcd}
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[rr, "\Pi"] \arrow[d, equal] & & \mathbb{V}_{(A,B),(B,C)} \arrow[d, "\Pi"'] & \mathbb{V}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] \arrow[d, "\Pi"'] & \mathbb{V}_{A,C} \arrow[d, "\Pi"] \\
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{(A,B),(B,C)} & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{P}_{A,C}
\end{tikzcd}$$

Notice that it does not make sense to ask whether this diagram commutes, since there is no starting or ending point in it. However, when it is lifted to categories of presheaves (where the Δ, $\sum$, and $\prod$ labels show how to lift functors), all Δ arrows are reversed, and the diagram turns into a diagram for which it makes sense to ask whether it commutes or not. When we lift this diagram to presheaf categories, the bottom row corresponds to taking two innocent strategies and composing them, so if the diagram (lifted to presheaf categories) commutes, then the composite of any strategies is in the image of $\prod_{\mathsf{i}_{A,C}}$, and is thus innocent. In the lifted diagram, the left-hand square commutes because the underlying one does, and the middle square commutes because the underlying one is exact. We thus only have to check that the right-hand one commutes, which is more difficult. Basically all the proofs in this chapter follow a common pattern. We study the underlying diagram and

- for all squares that are made exclusively of Δ (resp. $\prod$, resp. $\sum$), we show that the square commutes,

- for all squares that are of the form

$$\begin{tikzcd}
A \arrow[r, "\Pi"] & B \\
C \arrow[u, "\Delta"] \arrow[r, "\Pi"'] & D \arrow[u, "\Delta"']
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "\Sigma"] & B \\
C \arrow[u, "\Delta"] \arrow[r, "\Sigma"'] & D \arrow[u, "\Delta"']
\end{tikzcd}$$

  we show that the square is exact, which entails that the lifted square commutes up to isomorphism,

- for squares of the form

$$\begin{tikzcd}
A \arrow[r, "\Sigma"] \arrow[d, "\Pi"'] & B \arrow[d, "\Pi"] \\
C \arrow[r, "\Sigma"'] & D
\end{tikzcd}$$

  we have to rely on more complex proofs.

Chapter 2

Preliminaries

In this chapter, we give a list of results from the state of the art that will be used in this dissertation. In this thesis, each section will be prefaced by a list of the preliminary sections that the reader should have read to understand the current section.

2.1 Game Semantics

We will be mostly interested in a particular version of game semantics, called *arena games*. The idea, in all these models, is to interpret types as *arenas* and programs as *strategies* on arenas of the right type. The definitions of these game models mostly follow the same pattern: they first define their notion of arena, which usually describe all the possible ways a program of a given type may perform a single reduction step, then *plays*, which describe a particular interaction between a program and an environment, and finally *strategies*, which usually consist of a set of accepted plays.

Being able to compose strategies is an important part of game semantics, since this is why game semantical models are compositional: the strategy associated to the composite of two functions is the composite of the strategies associated to those functions. To define composition of strategies, all these models define *interaction sequences*, which basically represent the interaction of two plays. Finally, to show that composition of strategies is associative, they define *generalised interaction sequences*, which represent the interaction of three plays.

In Section 2.1.1, we study a game model called HON games. We then study a few possible variations of HON games, which are obtained by slightly changing the notion of play, in Section 2.1.2. Finally, we study Tsukada and Ong's games, which are another variation of HON games, in which the notion of morphism is different, in Section 2.1.3. In Section 2.1.4, we study AJM games, which are another game model. Finally, we study Blass games in Section 2.1.5, which are games that are well-known for their non-associative composition.

In Chapter 5, we will be interested in Tsukada and Ong's games, and their links to string diagrammatic models. In Chapter 6, we will be interested in all the game models we describe in this section.

2.1.1 Hyland-Ong/Nickau Games

Required: ∅.
Recommended: ∅.

For this model and a few others, the presentation we adopt is inspired by Harmer's PhD thesis [46], because it unifies and simplifies many different frameworks.

Definition 2.1.1. *An* arena *is a triple* $(A, \lambda, \vdash)$ *where* A *is a set,* $\lambda : A \to \{P, O\} \times \{!, ?\}$ *is a function, and* $\vdash \subseteq A \times A$ *is a relation that verifies:*

- *for all* m *in* A*, if there is no* n *such that* $n \vdash m$*, then* $\lambda(m) = (O, ?)$*,*
- *for all* n *and* m *in* A *such that* $m \vdash n$*,* $\lambda^{OP}(m) \neq \lambda^{OP}(n)$*, where* λ^{OP} *is defined as* $\pi_1\lambda$*,*
- *for all* n *and* m *in* A *such that* $m \vdash n$*, if* $\lambda^{?!}(n) = !$*, then* $\lambda^{?!}(m) = ?$*, where* $\lambda^{?!}$ *is defined as* $\pi_2\lambda$*.*

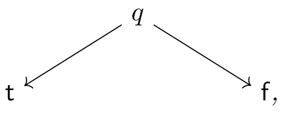
Terminology 2.1.2. *In an arena* $(A, \lambda, \vdash)$*, the elements of* A *are called* moves*,* λ *is called the* labelling *function, and* $\vdash$ *is called the* enabling *relation. A move* m *is said to* enable *a move* m' *if* $m \vdash m'$*. The set of all moves of* A *will be denoted by* M_A*, or sometimes simply* A*.*

A move is initial *(and is called a* root *of* A*) if there is no move that enables it. The set of roots of* A *is denoted* $\sqrt{A}$*.*

A move m *is an* Opponent *move when* $\lambda^{OP}(m) = O$*, and a* Proponent *move otherwise. A move* m *is a* question *when* $\lambda^{?!}(m) = ?$*, and an* answer *otherwise. In these terms, the constraints verified by* λ *state that all roots of* A *are Opponent questions, that the enabling relation alternates between Opponent and Proponent, and that answers are always enabled by questions.*

Remark. *Most of the settings we will study use exactly the same notion of arena, and only differ from HON game semantics because they impose different constraints on plays. AJM games and Blass games use different notions of arenas. There is also one model based on HON games that uses a different notion of arena: Tsukada and Ong's games, in which the enabling relation forms a forest.*

Example 2.1.3. *The boolean arena* $\mathbb{B}$ *comprises three moves:* q*, which is an Opponent question, and* t *and* f*, which are Proponent answers. It may be drawn as a graph:*



where the arrow denotes the enabling relation. This representation may be augmented with annotations for questions and answers to fully represent $\mathbb{B}$*, but since we will not need to be very formal about arenas, except in the case of Tsukada and Ong, who do not use questions and answers, we decide not to write them for conciseness.*

To model the interaction between a program fragment and its environment, game semantics uses sequences of moves in an arena. For example, the following sequence is an interaction between a program fragment computing a boolean, and its environment:

$$q \; \mathsf{t} \; q \; \mathsf{f}.$$

The first move corresponds to the environment asking for the value of the boolean. The second one corresponds to the program fragment answering that question by telling the environment that the boolean is true. The third and fourth moves play the same roles as the two previous moves, except that the program fragment answers that the boolean is false. This program fragment could, for example, be incrementing a reference each time it is called upon and answer true or false depending on the parity of that reference.

Obviously, some sequences of moves do not make sense. For instance, answering the value of a boolean before the environment even asked for that value makes little sense. Interactions are therefore represented by *justified sequences*, which prohibit such patterns by requesting that all moves be "justified" by some previous move.

Definition 2.1.4. *A* justified sequence *on an arena* $(A, \lambda, \vdash)$ *is a triple* (n, f, φ) *where* n *is a natural number,* f *is a map from* n *to* A *and* φ *is a map from* n *to* $\{0\} \uplus n$*, such that, for all* $i \in n$*,*

- $\varphi(i) < i$*,*
- *if* $\varphi(i) = 0$*, then* $f(i) \in \sqrt{A}$*, and*
- *if* $\varphi(i) \neq 0$*, then* $f(\varphi(i)) \vdash f(i)$*.*

Justified sequences equipped with prefix ordering form a category $\mathbb{P}_A$*.*

Remark. *The notation* $\mathbb{P}_A$ *for justified sequences on* A *is non-standard. Usually, plays on an arena* A *are defined like what we call plays on a pair of arenas in this dissertation (see below). We however prefer this definition because:*

- *it allows us to consider projections* $\mathbb{P}_{A,B} \to \mathbb{P}_A$ *and* $\mathbb{P}_{A,B} \to \mathbb{P}_B$*, which is impossible with the traditional notion of play on* A*,*
- *but it does not influence our study of strategies, since we only consider strategies built on* $\mathbb{P}_{A,B}$*'s (see below) and not* $\mathbb{P}_A$*'s.*

Terminology 2.1.5. *In a justified sequence* (n, f, φ)*, if* $\varphi(i) = j$*, we say that* j justifies i*, or that* j is the justifier of i*. The arrow that points from* i *to* j *is called a* justification pointer*.*

Justified sequences will be drawn as sequences of moves, to be read from left to right, with arrows pointing from each move to its justifier, as is classic in game semantics.

Our example above can thus be redrawn with justification pointers as:

$$\overset{\curvearrowleft}{q \; \mathsf{t}} \quad \overset{\curvearrowleft}{q \; \mathsf{f}}.$$

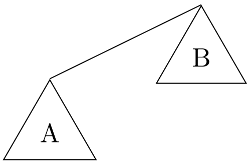
Game semantical frameworks then go on to define what a *play* is. A play represents a particular type of interaction between a function and an environment. In order to define them, we first need to define arenas for function types, which we do now.

To represent the interaction of a function of type $A \to B$ and its environment, we build an arrow arena $A \multimap B$ from the arenas A and B.

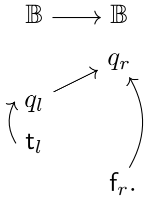
Definition 2.1.6. *The arena $A \multimap B$ is defined by*

- $M_{A \multimap B} = M_A + M_B$,
- $\lambda_{A \multimap B} = [\overline{\lambda_A}, \lambda_B]$, *where $\overline{\lambda}$ is $(s \times \{?, !\})\lambda$, where s is the swap function on $\{O, P\}$,*
- *and, for all m and m' in $M_{A \multimap B}$, $m \vdash_{A \multimap B} m'$ if and only if any of the following holds:*
  - $m \vdash_A m'$ *or* $m \vdash_B m'$, *or*
  - m *is in* $\sqrt{B}$ *and* m' *is in* $\sqrt{A}$.

If we draw arenas as trees, then we may draw $A \multimap B$ as:



In other words, $A \multimap B$ comprises the moves from A and those from B, with exactly the same structure, except that initial moves in A are enabled by initial moves in B and that polarity is reversed in A. An interaction in such an arena is thought of as a player who may play on two games: as Proponent in B and as Opponent in A (this is rendered explicit by the fact that polarity is reversed in A). To easily tell in which part of $A \multimap B$ a move is played, we draw such interactions as a sequence of moves to be read from top to bottom, with the moves of A played on the left and those of B played on the right, as in:



In this interaction, a player plays on two boolean games, one on the left and one on the right. They play as Proponent on the right-hand side game and as Opponent on the left-hand side one. For simplicity let us call this player M (for "middle"). There are two other players L and R who play Proponent on the left-hand game and Opponent on the right-hand game, respectively (they may also be thought of as a single player that represents the whole environment).

The drawing above represents a possible interaction of the `not` function on booleans with an environment.

- The first move of the interaction (q_r), which is played by the environment (represented by the player R on the right), corresponds to asking the function to compute its result.

- In order to compute that result, the function must know the value of its argument. It thus asks the value of that argument to its environment, and more precisely to another program fragment that is represented by the player L on the left, by playing q_l.
- After computing the value of that boolean, L answers the value of the boolean: the boolean is true, which is encoded in the fact that L plays t_l.
- Since M now knows the value of its argument, it can finally answer what its result is by playing f_r.

This example illustrates why there should be a change of polarity in A: when the function asks for the value of its argument, it plays the role of the environment to the program fragment that computes this argument, thus it should be playing as Opponent on A.

We may also sometimes write plays on arenas $A \multimap B$ like other plays, in which case we will write m_l or m_A if the move m is played in the left-hand side arena and m_r or m_B if it is played in the right-hand side one.

Now that we have defined arenas for functional types, we may come back to the definition of plays. The exact notion of play differs between frameworks, but they all amount to imposing constraints on justified sequences. We will use the following notion of play in this manuscript, as well as numerous variations on it, which we will describe later.

Definition 2.1.7. *A* play *on the pair of arenas (A, B) is a justified sequence (n, f, φ) on $A \multimap B$ that is:*

- alternating*: for all $i < n$, $\lambda^{OP}(f(i)) \neq \lambda^{OP}(f(i+1))$,*
- *of even length: n is even.*

Prefix ordering between plays is defined in the obvious way: (n, f, φ) is a prefix of (m, g, ψ) if and only if $n \leq m$ and, for all $i \leq n$, $f(i) = g(i)$ and $\varphi(i) = \psi(i)$.

Plays on the arena pair (A, B) equipped with prefix ordering form a category $\mathbb{P}_{A,B}$.

In the other variants of game semantics discussed below, the notion of play is modified by adding further constraints on them. In the case of Tsukada and Ong's model, however, there is a more fundamental difference between the categories of plays: their categories of plays have more morphisms than those simply given by prefix ordering.

A program fragment is then interpreted as a set of plays. The intuition is that a program fragment is interpreted as the set of interactions it can actually have with an environment. Such a set of plays is called a *strategy*.

Definition 2.1.8. *A* strategy *on the arena pair (A, B) is a prefix-closed set of plays on (A, B).*

Sometimes, strategies are also asked to be non-empty (i.e., to contain at least the empty play). A strategy σ is said to *accept* the play s if $s \in \sigma$ and to *reject* it otherwise.

Prefix-closedness comes from the fact that, if a program and its environment may interact, then they can have exactly the same interaction, but cut at some point, so if a strategy accepts a play, it should also accept all its prefixes.

Example 2.1.9. *Let us consider the* `not` *function on booleans. It may be interpreted as a strategy on the arena* $\mathbb{B} \multimap \mathbb{B}$ *that accepts all plays* $s = (2n, f, \varphi)$ *such that, for all* i *in* n*:*

- *if* $f(2i-1) = q_B$, *then* $\begin{cases} f(2i) = q_A \\ \varphi(2i) = 2i-1 \end{cases}$,
- *if* $f(2i-1) = \mathsf{t}_A$, *then* $\begin{cases} f(2i) = \mathsf{f}_B \\ \varphi(2i) = \varphi(2i-1) - 1 \end{cases}$, *and similarly if* $f(2i-1) = \mathsf{f}_A$.

This illustrates the behaviour of the `not` *function:*

- *whenever the environment asks for the value of* `not` *on a boolean (when there is a* q_B *move), the strategy asks for its argument (by playing* q_A*) and remembers which question it is trying to answer, which is represented by the justification pointer pointing to a* q_B*,*
- *whenever it receives a value for its argument (either* t_A *or* f_A*), it answers the environment's corresponding* q_B *question with the right value (either* f_B *or* t_B*); to know which* q_B *question it should answer, the strategy looks at which of its* q_A *question was answered (by looking at the justification pointer of the* t_A *or* f_A *move) and answers the corresponding* q_B *question, which necessarily comes just before the* q_A *question.*

A nice feature of game semantics is that it may interpret different classes of functions depending on the restraints imposed on strategies. For example, the strategy in the example above is very constrained, in the sense that Proponent may not do much: the move it plays only depends on the previous move and its justification pointer. In other words, its reaction only depends on the current "call to the function". Technically, this strategy is *innocent* (this term will be defined later). Such strategies may only interpret purely functional programs.

Here is an example of strategy that is not innocent:

Example 2.1.10. *Let us consider a boolean function that keeps a reference to an integer, increments it each time it is called, and returns either its argument if the reference is even, or its negation if the reference is odd. This function is not purely functional, since it modifies the value of a reference. The strategy associated to such a function could, for example, be the strategy* σ *that accepts all plays* $s = (n, f, \varphi)$ *such that, for all* $i \in n$*:*

- *if* $f(2i-1) = q_B$, *then* $\begin{cases} f(2i) = q_A \\ \varphi(2i) = 2i-1 \end{cases}$,
- *if* $f(2i-1) = b_A$ *and the number of* q_B *moves in* $s_{|2i-1}$ *is odd, then* $\begin{cases} f(2i) = \text{not } b_B \\ \varphi(2i) = \varphi(2i-1) - 1 \end{cases}$, *and similarly when the number of* q_B *moves is even, with* $f(2i) = b_B$.

This strategy indeed has the expected behaviour, since it answers either as the `not` *function or as the identity, depending on how many times it has been called (given by the number of* q_B *moves). Notice how this strategy no longer depends only on the current call to the function, but also on how many times the function has been called in total (this means that the strategy is not* innocent*).*

We want game semantics to be compositional, i.e., the strategy associated to the composite of two programs should be the composite of the strategies associated to those programs. In order for this to be the case, we should give a definition of how to compose strategies.

To compose two strategies $A \to B$ and $B \to C$, the idea is to let them interact on B and then to hide that interaction, leaving only the parts on A and C.

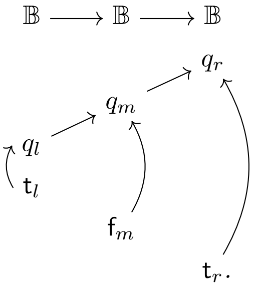
But we first need to define the projection of a justified sequence s on the arena $(\ldots(A_1 \multimap A_2) \multimap \ldots) \multimap A_n$ to the arena $(\ldots(((\ldots(A_1 \multimap A_2)\ldots) \multimap A_{i-1}) \multimap A_{i+1})\ldots) \multimap A_n$. It is defined as the justified sequence with the same moves as s, except for the moves in A_i, and justification pointers inherited from those of s: the only case where this does not make sense is when an initial move m_{i-1} in A_{i-1} points to an initial move m_i in A_i, but then m_i points to an initial move m_{i+1} in A_{i+1}, so we define m_{i-1} to point to m_{i+1} in the projection.

To define such an interaction, we need to study "justified sequences on three arenas".

Definition 2.1.11. *An* interaction sequence *on the arena triple (A, B, C) is a justified sequence on $(A \multimap B) \multimap C$ such that its projections to $A \multimap B$ and $B \multimap C$ are plays and whose last move is either in A or C.*

Interaction sequences on the arena triple (A, B, C) equipped with prefix ordering form a category $\mathbb{P}_{A,B,C}$.

Example 2.1.12. *Here is an example of an interaction sequence on $(\mathbb{B}, \mathbb{B}, \mathbb{B})$, to be read from top to bottom:*


$\mathbb{B} \longrightarrow \mathbb{B} \longrightarrow \mathbb{B}$
q_r
q_m
q_l
$\mathtt{t}_l$
$\mathtt{f}_m$
$\mathtt{t}_r$.

This could for example represent a call to the function `not` ∘ `not` *on the argument* `t`*:*

- *q_r corresponds to the function being called,*
- *then q_m to the function calling its argument, i.e., calling the* `not` *function once again,*
- *then the second call to* `not` *calls its argument by playing q_l,*
- *the value of the argument is* `t`*, which is computed directly and sent back to the second call to* `not` *by the move $\mathtt{t}_l$,*
- *now that it knows the value of its argument, the second call to* `not` *can return the answer* `f` *by playing $\mathtt{f}_m$,*
- *finally, the first call to* `not` *can now answer by playing $\mathtt{t}_r$.*

It is an interaction sequence because its projections are plays and its last move is in the right-hand side arena.

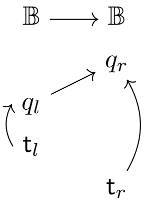
We now have all the tools necessary to define composition of strategies:

Definition 2.1.13. *The composite $\sigma;\tau$ of two strategies σ on $A \multimap B$ and τ on $B \multimap C$ accepts a play s if and only if there is an interaction sequence u:*

- *whose projection to $A \multimap C$ is s,*
- *and such that its projection to $A \multimap B$ is accepted by σ and its projection to $B \multimap C$ is accepted by τ.*

Let us see how the composite $\sigma;\tau$ may be seen as σ and τ interacting on B, then hiding what happens on B from the result of that interaction. The traditional way to define composition is in two steps, as described above. The first step, in which the two strategies interact, is called *parallel composition*, and the second one, in which we hide what happens on B, is called *hiding.* The parallel composition $\sigma \| \tau$ of σ and τ is a "strategy" on $\mathbb{P}_{A,B,C}$, i.e., a prefix-closed set of interaction sequences, that accepts an interaction sequence u if and only if there are plays s and t accepted by σ and τ respectively and that can "interact" on B to combine into u. Hiding then just literally hides what happens on B, slightly reorganising the justification pointers from A to B to make them point to C (note that this is very simple, since pointers from moves in A to moves in B necessarily point to initial moves in B, which themselves point to initial moves in C).

Example 2.1.14. *For example, if we call σ the strategy associated to the* `not` *function, the composite of σ with itself accepts*



because the interaction sequence in Example 2.1.12 is equal to this play when projected to $A \multimap C$ (we use A, B, and C here to distinguish between the three $\mathbb{B}$ arenas), and its projections to $A \multimap B$ and $B \multimap C$ are accepted by σ. Notice also how this indeed corresponds to how the composite of the `not` *function with itself may behave: indeed, this composite is simply the identity function, which does return* `t` *when its argument is* `t`*.*

Now that we know how to compose strategies, the next step is to organise them into a category. We thus see strategies on $A \multimap B$ as maps from A to B. For arenas and strategies to form a category, composition of strategies needs to be associative and unital.

The classical way to prove that composition of strategies is associative is to define *generalised interaction sequences*, which are justified sequences on four arenas.

Definition 2.1.15. *A* generalised interaction sequence *on the tuple* (A,B,C,D) *of arenas is a justified sequence on* $((A \multimap B) \multimap C) \multimap D$ *whose projections to* $A \multimap B$*,* $B \multimap C$*, and* $C \multimap D$ *are plays, and whose last move is either in* A *or* D*.*

Generalised interaction sequences on the tuple of arenas (A,B,C,D) *form a category* $\mathbb{P}_{A,B,C,D}$ *when equipped with prefix ordering.*

The crucial lemma [9] to prove that composition of strategies is associative is the following:

Lemma 2.1.16 (Zipping Lemma)**.** *The two squares*

$$\begin{tikzcd}
\mathbb{P}_{A,B,C,D} \arrow[r] \arrow[d] & \mathbb{P}_{A,C,D} \arrow[d] \\
\mathbb{P}_{A,B,C} \arrow[r] & \mathbb{P}_{A,C}
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{P}_{A,B,C,D} \arrow[r] \arrow[d] & \mathbb{P}_{A,B,D} \arrow[d] \\
\mathbb{P}_{B,C,D} \arrow[r] & \mathbb{P}_{B,D}
\end{tikzcd}$$

are pullbacks on objects.

Once we know that composition of strategies is associative, we also want to know that it is unital. The units are the *copycat* strategies, which accept only *copycat* plays.

Definition 2.1.17. *A play* $s = (2n, f, \varphi)$ *on* (A,A) *is* copycat *if, for all* i *in* n*,* $f(2i) = \overline{f(2i-1)}$ *and* $\varphi(2i) = \varphi(2i-1) - 1$*, where* $\overline{m_r} = m_l$ *and* $\overline{m_l} = m_r$*.*

Copycat plays simply copy whatever the environment does on one copy of A to the other one.

Example 2.1.18. *The play illustrated in Example 2.1.14 is a copycat play.*

Definition 2.1.19. *The* copycat strategy *on* (A,A) *is the strategy that exactly accepts copycat plays.*

We may now form a category of arenas and strategies with:

- arenas as objects,
- strategies on (A,B) as morphisms from A to B,
- composition of strategies as composition,
- copycat strategies as identities.

One can then show that this category is symmetric monoidal closed, impose some more conditions on plays (as we see in the next section) to make it cartesian closed and interpret λ-calculus into it, but we will not give any details, as this will not be relevant to us.

However, the following point will be very relevant to us. A class of strategies that we will be particularly interested in is the class of *innocent* strategies. They are important because they correspond to functional programs that do not use references (stateless programs).

Basically, a strategy is innocent if it decides to accept plays based not on the whole history of the interaction it has had with the environment, but only on restricted parts of it. The intuition is that the parts of the play the strategy may depend on are the parts which “led” to the current move. Such a part is called a *view*.

Definition 2.1.20. *For all plays $s = (n, f, \varphi)$, and i in n, the* Proponent view *(also simply* view*) of s at index i, denoted by $\ulcorner s \urcorner_i$ is the sequence defined inductively by:*

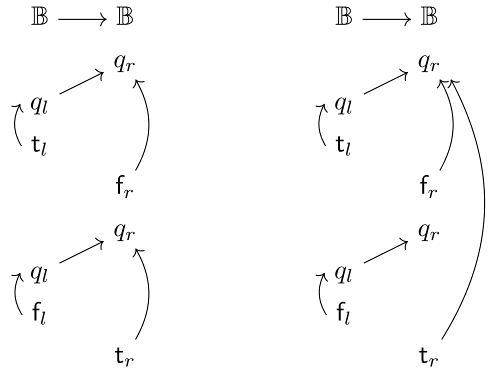
- $\ulcorner s \urcorner_i = f(i)$ *if* $\varphi(i) = 0$,
- $\ulcorner s \urcorner_i = \ulcorner s \urcorner_{\varphi(i)} \cdot f(i)$ *if* $f(i)$ *is an Opponent move (i.e., i is odd) and* $\varphi(i) \neq 0$,
- $\ulcorner s \urcorner_i = \ulcorner s \urcorner_{i-1} \cdot f(i)$ *if* $f(i)$ *is a Proponent move (i.e., i is even).*

First of all, if we want to be exact, $\ulcorner s \urcorner_i$ is not a sequence of moves, but a subsequence of indices of s, equipped with f inherited from s. Furthermore, note that $\ulcorner s \urcorner_i$ is not an ordinal, but is included in ω, so it is canonically isomorphic to a single ordinal (which is the definition of view, if we want to be more formal).

Note that, in general, $\ulcorner s \urcorner_i$ is a mere sequence, and not a justified sequence. For this sequence to be a justified sequence, the following condition must be satisfied by the play:

Definition 2.1.21. *A play $s = (n, f, \varphi)$ is P-*visible *if each Proponent move is justified by a move in its view. In other words, for all even i in n, $\varphi(i) \in \ulcorner s \urcorner_i$.*

Example 2.1.22. *Let us consider the play s on the left and t on the right below.*



Its view $\ulcorner s \urcorner_8 = \ulcorner s \urcorner_7 \cdot \mathsf{t}_r = \ulcorner s \urcorner_6 \cdot \mathsf{f}_l \mathsf{t}_r = \ulcorner s \urcorner_5 \cdot q_l \mathsf{f}_l \mathsf{t}_r = q_r q_l \mathsf{f}_l \mathsf{t}_r$. So t_r is justified by q_r, which is in its view. If we repeat this process for all Proponent moves, we get that s is P-visible. On the other hand, t is not P-visible, because $\ulcorner s \urcorner_8$ does not contain the first occurrence of q_r, which is the justifier of t_r.

The moves an innocent strategy σ may play after a play s depend only on the view $\ulcorner s \urcorner_n$. Since what an innocent strategy may do only depends on views, whether it accepts a play or not only depends on its views. We thus want to define innocent strategies as strategies that accept or reject views rather than plays.

Definition 2.1.23. *A* view *on the pair of arenas (A, B) is a non-empty play $s = (n, f, \varphi)$ such that $\ulcorner s \urcorner_n = s$.*

Views equipped with prefix ordering form a category $\mathbb{V}_{A,B}$.

Views are exactly the plays that arise as $\ulcorner s \urcorner_i$ for some P-visible play $s = (n, f, \varphi)$ and i in n. The following characterisation of views is also sometimes useful:

Lemma 2.1.24. *A play $s = (n, f, \varphi)$ is a view if and only if $n \neq 0$ and all Opponent moves are justified by their predecessors, or in other words, for all odd i in n, $\varphi(i) = i - 1$.*

We now define innocent strategies as a set of accepted views. Except that we do not call these innocent strategies, but *behaviours*, as we reserve the term *strategy* for sets of accepted plays.

Definition 2.1.25. *A* behaviour *on the pair of arenas (A, B) is a prefix-closed set of views on (A, B).*

Just like for plays, we say that a behaviour σ *accepts* a view v when $v \in \sigma$ and that it *rejects* it otherwise.

We may now define innocent strategies:

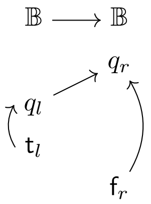
Definition 2.1.26. *A strategy σ on (A, B) is* innocent *if it only accepts P-visible plays and, for all plays s, σ accepts s if and only if it accepts all its views $\ulcorner s \urcorner_i$.*

There is an isomorphism between behaviours and innocent strategies:

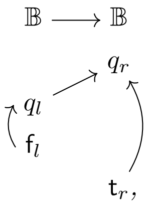
- to each behaviour σ, we may associate the strategy that accepts all plays whose views are accepted by σ,
- to each strategy σ, we may associate the behaviour that accepts all views accepted by σ,

because an innocent strategy accepts a play if and only if it accepts all its views, the mappings above are indeed isomorphisms.

Example 2.1.27. *The strategy of Example 2.1.9 is innocent. Indeed, the behaviour associated to that strategy is the behaviour that accepts all views of the form*

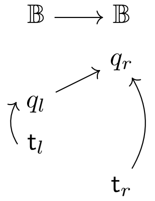


and

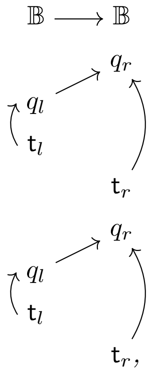


and their (non-empty) prefixes of even length. The innocent strategy associated to that behaviour is exactly the strategy of Example 2.1.9, which is therefore innocent.

However, the strategy of Example 2.1.10 is not innocent. Indeed, the behaviour associated it accepts the view

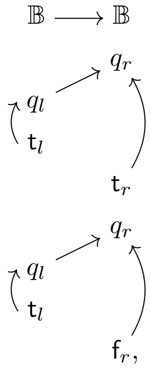


among others, because the strategy accepts it. But then, the strategy associated to that behaviour must accept

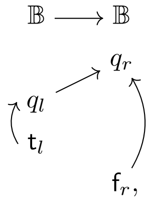


which the original strategy does not accept, so the original strategy is not innocent.

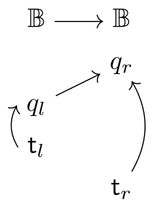
The reader may note that the strategy obtained in the second part of Example 2.1.27 is non-deterministic. Indeed, the original strategy accepts the play



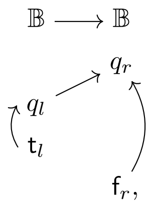
so the behaviour associated to it should accept



which is one of its views. But then, the behaviour accepts both



and



so it is clearly non-deterministic. A typical constraint that can be imposed on strategies is determinism:

Definition 2.1.28. *A strategy σ is* deterministic *if, for all plays s, if sab and sab' are accepted, then $b = b'$.*

More precisely, not only should b be equal to b' in the definition above, but they should also point to the same move. Deterministic strategies can obviously only interpret deterministic programs.

A crucial result of game semantics is that deterministic innocent strategies compose. More precisely:

Lemma 2.1.29 (Preservation of innocence)**.** *If $\sigma: A \to B$ and $\tau: B \to C$ are two deterministic innocent strategies, then the composite $\sigma;\tau$ as defined in Definition 2.1.13, is again deterministic and innocent.*

Moreover, copycat strategies are easily shown to be innocent and deterministic. We thus get:

Lemma 2.1.30. *Deterministic, innocent strategies form a subcategory of the category of arenas and strategies.*

2.1.2 Variations on HON Games

Required: 2.1.1.
Recommended: $\varnothing$.

As we alluded to before, a nice point of HON-style game semantics is that there are many variations on it, and that these slight variations give models of different features of programming languages. Since the goal of Chapter 6 is to define a general framework to study game semantics, we should show that all these different variations can be accounted for. We thus define a list of different HON-style game settings, most of them done again in the style of Harmer's PhD thesis [46] and the last one based on McCusker's PhD thesis [78].

Constraints on Plays

Except for innocence and determinism, which we have already defined, all these variations are defined similarly: we first restrict the notion of play by asking that they verify additional constraints, then define strategies as prefix-closed sets of plays (for the new notion of play), and finally verify that this new notion of strategy forms a subcategory of the initial notion of strategy.

There are four main constraints that may be imposed on plays: *P-visibility* (which we have already discussed), *O-visibility* (which is the counterpart to P-visibility for Opponent), *well-bracketing*, and *well-threadedness.* Each set of constraints taken among those gives a new notion of strategy. Let us thus describe the different constraints we have not described yet.

Let us start with *O-visibility.* Just like P-visibility, we first need to define a notion of view, but this time for Opponent, rather than Proponent, which is exactly dual to the definition of Proponent views, though it must be expressed slightly differently because of the only dissymmetry between Opponent and Proponent: initial moves are always played by Opponent.

Definition 2.1.31. *For all plays $s = (n, f, \varphi)$, and i in n, the* Opponent view *(or O-*view*) of s at index i, denoted $\lfloor s \rfloor_i$ is the sequence defined inductively by:*

- $\lfloor \varepsilon \rfloor_i = \varepsilon$,
- $\lfloor s \rfloor_i = \lfloor s \rfloor_{\varphi(i)} \cdot f(i)$ *if $f(i)$ is a Proponent move (i.e., i is even),*
- $\lfloor s \rfloor_i = \lfloor s \rfloor_{i-1} \cdot f(i)$ *if $f(i)$ is an Opponent move (i.e., i is odd).*

Like in the case of P-visibility, the O-view is not necessarily a justified sequence, but it is when the play verifies a condition dual to P-visibility:

Definition 2.1.32. *A play $s = (n, f, \varphi)$ is O-*visible *if each Opponent move is justified by a move in its O-view. In other words, for all odd i in n, $\varphi(i) \in \lfloor s \rfloor_i$.*

Let us now describe *well-bracketing.* Well-bracketed strategies can exactly interpret programs that do not contain control operators (such as *call/cc*). This is done by forcing such strategies to always answer the last pending question asked by the environment. To define this, we first need to define pending questions:

Definition 2.1.33. *A question i is the* pending question *in $s = (n, f, \varphi)$ if it is the last Opponent question in s that justifies no answer, or in other words:*

- $\lambda(f(i)) = (O, ?)$,
- *for all j in n, if $\varphi(j) = i$, then $\lambda^{?!}(f(j)) = ?$,*
- *for all $j > i$ in n, if $\lambda(f(j)) = (O, ?)$, then there is a k in n such that $\varphi(k) = j$ and $\lambda(f(j)) = (P, !)$.*

Definition 2.1.34. *A play $s = (n, f, \varphi)$ is* well-bracketed *if all Proponent answers answer the last pending Opponent question in their view, or in other words, for all even i in n, if $f(i)$ is a Proponent answer, then $\varphi(i)$ is the pending question of $\lceil s \rceil_i$.*

Let us finally describe *well-threadedness.* This constraint is more general than P-visibility, as Proponent moves are not required to point into their views, but only into their *threads*:

Definition 2.1.35. *A move i in $s = (n, f, \varphi)$ is* hereditarily justified *by j if $i = j$ or $\varphi(i)$ is hereditarily justified by j.*

If $s = (n, f, \varphi)$ is a justified sequence, the thread *of the ith move, denoted $[s]_i$, is the sequence of all indices j that are justified by the initial move that is a hereditary justifier of i. This notion can be generalised: if I is a set of indices in n, then $[s]_I$ is the union of all sub-sequences $[s]_i$, for i in I.*

A play $s = (n, f, \varphi)$ is well-threaded *when all Proponent moves point inside their predecessors' threads, or in other words, for all even i in n, $\varphi(i) \in [s]_{i-1}$.*

With all these different constraints, we can define new notions of plays:

Definition 2.1.36. *Let $C = \{P\text{-}vis, w\text{-}b, w\text{-}t\}$ be the set of constraints defined above (except for O-visibility). For any subset c of C, the category $\mathbb{P}^c_{A,B}$ of* c-plays *(or simply* plays *when c is clear from context) on (A, B) is the full subcategory of $\mathbb{P}_{A,B}$ spanning plays that verify all constraints in c.*

Example 2.1.37. *If we take $c = \varnothing$, then we get the previous notion of play.*

If we take $c = \{P\text{-}vis\}$, then we get the set of P-visible plays. This is the notion of plays used by Tsukada and Ong, but they are equipped with a different notion of morphism, so we will treat them separately later.

If we take $c = C$, then we get the notion of P-visible, well-bracketed plays (because P-visibility implies well-threadedness).

For all these different notions of plays, we can define notions of strategies:

Definition 2.1.38. *For any subset c of C, a* c-strategy *is a prefix-closed set of c-plays.*

Of course, we can then define innocent and deterministic strategies just like in the standard case:

Definition 2.1.39. *A c-strategy is* innocent *when it only accepts P-visible c-plays and accepts a c-play s if and only if it accepts all the views of s.*

A c-strategy is deterministic *if, when it accepts c-plays sab and sab', then $b = b'$.*

To define composition of strategies, we must then define interaction sequences:

Definition 2.1.40. *Let c be a subset of C. The category $\mathbb{P}^c_{A,B,C}$ of* c-interaction sequences *is the subcategory of $\mathbb{P}_{A,B,C}$ spanning interaction sequences whose projections to $\mathbb{P}_{A,B}$ and $\mathbb{P}_{B,C}$ actually fall into $\mathbb{P}^c_{A,B}$ and $\mathbb{P}^c_{B,C}$ respectively.*

For any set of constraints c, we get that the projection from $\mathbb{P}^c_{A,B,C}$ to $\mathbb{P}_{A,C}$ actually falls into $\mathbb{P}^c_{A,C}$.

The definition of composition then just mimics that explained in the previous section:

Definition 2.1.41. *If σ is a strategy on (A,B) and τ a strategy on (B,C), then the* composite *strategy $\sigma;\tau$ is the strategy on (A,C) that accepts a play s if and only if there is an interaction sequence u in $\mathbb{P}^c_{A,B,C}$:*

- *whose projection to $\mathbb{P}^c_{A,C}$ is s,*
- *and such that its projection to $\mathbb{P}^c_{A,B}$ is accepted by σ and its projection to $\mathbb{P}^c_{B,C}$ is accepted by τ.*

Finally, to study associativity of composition for such strategies, we must define generalised interaction sequences:

Definition 2.1.42. *Let c be a subset of C. The category $\mathbb{P}^c_{A,B,C,D}$ of* c-generalised interaction sequences *is the subcategory of $\mathbb{P}_{A,B,C,D}$ spanning generalised interaction sequences whose projections to $\mathbb{P}_{A,B}$, $\mathbb{P}_{B,C}$, and $\mathbb{P}_{C,D}$ actually fall into $\mathbb{P}^c_{A,B}$, $\mathbb{P}^c_{B,C}$, and $\mathbb{P}^c_{C,D}$ respectively.*

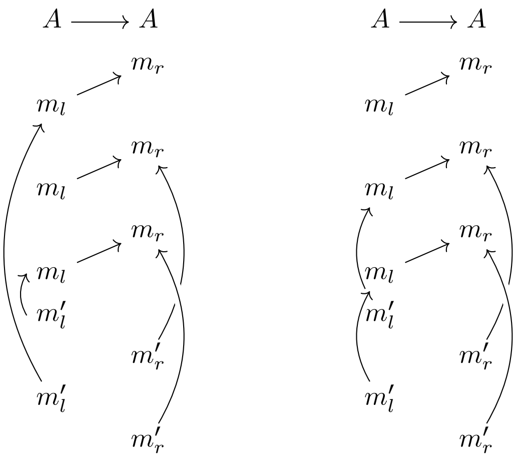
The crucial lemma to prove associativity of strategies is once again the zipping lemma, whose proof is obvious, since the zipping lemma is true for the basic notion of play, and being a c-interaction sequence or c-generalised interaction sequence is simply about projections being in $\mathbb{P}^c$'s, the constrained versions of $\mathbb{P}$'s.

We should finally define identities for these new notions of composition:

Definition 2.1.43. *A c-strategy on* (A, B) *is* copycat *if it accepts exactly copycat c-plays on* (A, B)*.*

We do not prove here that arenas and c-strategies form a subcategory of that of arenas and strategies because this point will be treated more elegantly in Chapter 6.

Let us mention the case of O-visibility. It is also well-known that P-visible, O-visible strategies (defined just like other c-strategies) form a category. However, strategies that are only O-visible do not form a category, which is why we treat O-visible strategies separately. Indeed, if O-visible strategies are to form a category, the identities must be the copycat strategies. The problem is that not all copycat plays are O-visible, which leads to "forgetting" some plays when composing with a copycat strategy. More precisely, let us consider an O-visible strategy σ on (A, A) that accepts the play p on the left below (notice that p is indeed O-visible).



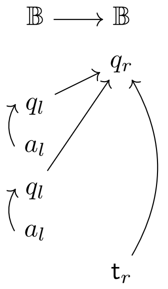
If we want p to be accepted by $\sigma; id_A$, then the copycat play on the right above should be accepted by id_A. But that play is not O-visible (the second m'_l move does not point into its O-view), so it cannot be accepted by id_A, which means that p cannot be accepted by $\sigma; id_A$, and id_A is thus not an identity.

One could also study other restrictions on plays, such as well-openedness:

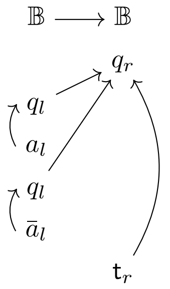
Definition 2.1.44. *A play is* well-opened *if there is at most one initial move in it.*

There is however a problem with this constraint, because composition cannot be defined as easily as in the cases studied above:

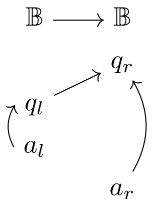
Example 2.1.45. *Let us consider a program of type* $\mathbb{B} \to \mathbb{B}$ *that calls its argument twice and answers* t *if the value of that argument is the same in each call, and* f *if the values are different. In the context of well-opened strategies, this program would be interpreted as the strategy* σ *that accepts all plays of the form*



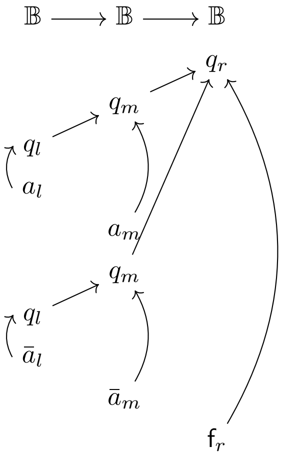
and

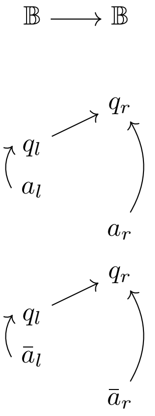


for all booleans a, and all their prefixes. Let us also consider the identity function on booleans, which is interpreted as the strategy id that accepts all plays of the form



for all booleans a, and their prefixes. Intuitively, when we compose both strategies id;σ, we should obtain σ again. However, if we take an interaction sequence that should intuitively correspond to the interaction of these two programs, we obtain the interaction sequence on the left below, which, when projected to the left-hand side arenas, gives the justified sequence on the right below.





But this play is not accepted by σ, since it is not even well-opened. This example shows that composition of well-opened plays is somewhat more involved than in the cases that we have treated, as we need to define a ! (bang) modality that corresponds to repeating.

Further Constraints on Plays

Another possible variation on HON games is described in McCusker's PhD thesis [78]. The idea is that we impose a further constraint on the set of plays.

The terminology for plays is slightly different from that of the previous sections because there are now two notions of plays: the first one (*legal plays*) is constrained only by conditions we have already mentioned, and the other one (simply *plays*) is constrained by a further predicate.

Definition 2.1.46. *A* legal play *(which McCusker calls* legal positions*) on the arena A is an alternating, P-visible, O-visible, well-bracketed justified sequence of even length on A. The set of legal plays on A is L_A.*

In terms of vocabulary from the previous section, a legal play is basically a C-play.

We then choose a predicate that will tell which legal plays are actual plays.

Definition 2.1.47. *An* arena with predicate *is a pair (A, P_A) of an arena A and a predicate P_A on L_A such that:*

- *P_A is non-empty (there is s in L_A such that $P_A(s)$ is true),*
- *P_A is prefix-closed (if $P_A(sab)$ is true, then so is $P_A(s)$),*
- *P_A is stable under "generalised threads" (if $P_A(s)$ is true and I is a set of indices in s, then $P_A([s]_I)$ is also true).*

A play *(which McCusker calls a* valid position*) on (A, P_A) is a legal play s on A such that $P_A(s)$ is true.*

We will sometimes denote the arena with predicate (A, P_A) simply by A, leaving the context to disambiguate.

Finally, a play on a pair of arenas is defined as a legal play whose projections respect the constraints imposed by the predicates:

Definition 2.1.48. *For all arenas with predicates (A, P_A) and (B, P_B) the* arrow *arena with predicate is defined as $(A \multimap B, P_{A \multimap B})$, where $P_{A \multimap B}(s)$ holds if and only if P_A holds on the projection of s to A and P_B holds on its projection to B. A* play *on the pair of arenas with predicates $((A, P_A), (B, P_B))$ is a play on $(A \multimap B, P_{A \multimap B})$.*

For all such pairs of arenas with predicates $((A, P_A), (B, P_B))$, plays on the pair of arenas (A, B) equipped with prefix ordering form a category $\mathbb{P}^P_{A,B}$.

Interaction sequences and generalised interaction sequences are then defined as before:

Definition 2.1.49. *An* interaction sequence *on the tuple of arenas (A, B, C) is a legal play of $(A \multimap B) \multimap C$ whose left and right projections are in $\mathbb{P}^P_{A,B}$ and $\mathbb{P}^P_{B,C}$ respectively and whose last move is either in A or C. Interaction sequences on (A, B, C) equipped with prefix ordering form a category $\mathbb{P}^P_{A,B,C}$.*

A generalised interaction sequence *on the tuple of arenas (A, B, C, D) is a legal play of $((A \multimap B) \multimap C) \multimap D$ whose projections are in $\mathbb{P}^P_{A,B}$, $\mathbb{P}^P_{B,C}$, and $\mathbb{P}^P_{C,D}$ respectively and whose last move is either in A or D. Generalised interaction sequences on (A, B, C, D) equipped with prefix ordering form a category $\mathbb{P}^P_{A,B,C,D}$.*

Since the definition of strategy and composition of strategies are exactly the same as above, we spare the reader from having to read them yet another time. Copycat plays are simply plays (i.e., legal plays on which P holds) that are also copycat, and strategies are copycat if and only if they accept all copycat plays, so we omit their formal definitions. Once again, associativity of composition of strategies relies on a zipping lemma, which obviously holds because being a play or an interaction sequences is only about projections being legal. One can therefore prove that, in this setting, arenas and strategies form a category with copycat plays as identities and composition of strategies as composition.

2.1.3 Tsukada and Ong's Model

Required: 2.1.1.
Recommended: $\varnothing$.

Here is a brief recapitulation on Tsukada and Ong's categories of views and plays, as well as their notion of strategy. We will not use a specific vocabulary when talking about their notions of arenas, plays, etc, so there could be a slight ambiguity with the previous game models. However, the only part in which we will talk about the other settings is Chapter 6, and more precisely when we treat concrete examples, in which case the notions of arenas, plays, etc, will be clear from context.

As in the other game semantical settings, games are based on arenas. Here, however, arenas are forests (moves have at most one justifier).

Notation 2.1.50. *If A is an arena and m is a move in A, then $A_{/m}$ is the forest strictly below m, and $A \cdot m$ denotes $A_{/m}$ when $m \in \sqrt{A}$. If A is an arena, we denote by $m.A$ the tree t whose root is m and such that $t \cdot m = A$. Note that any arena is a coproduct of trees, so it can be written $A = \sum_{m \in \sqrt{A}} m.(A \cdot m)$.*

Remark. *This graph-based notion of arena, already used, e.g., by Harmer, Hyland, and Melliès [47], slightly differs from Tsukada and Ong's relation-based notion. For instance, their notion comprises non-empty arenas without initial moves, e.g., the integers with justification given by predecessor and ownership given by parity. But in fact, any relation-based arena A is isomorphic (in the category of arenas and innocent strategies) to the simple forest F_A given by the part of A which is reachable from its roots. Furthermore, F_A and A yield the exact same categories of views and plays. So we deliberately restrict attention to graph-based arenas.*

Let us fix arenas A and B. Let $A \multimap B$ denote the simple graph defined as

- $A + B$, with an edge $b \to a$ for all $b \in \sqrt{B}$ and $a \in \sqrt{A}$ if B is non-empty,
- the empty graph otherwise.

Even though $A \multimap B$ is not necessarily an arena, the notion of ownership straightforwardly extends to $A \multimap B$, since all paths from any root to some vertex v have the same length. Concretely, ownership is left unchanged in B but reversed in A. (This is the exact same construction as in HON games, and it is an arena in that setting.)

We denote by $|s|$ the length n of the sequence $s = (n, f, \varphi)$.

We will use the notion of preplay, which slightly generalises that of play. This is also where the real difference between HON games and Tsukada and Ong's games appear: the notion of morphism is different.

Definition 2.1.51. *A* TO-preplay *on the pair of arenas (A, B) is a P-visible, alternating, justified sequence on (A, B).*

A morphism of TO-preplays g from a TO-preplay (n, f, φ) on (A, B) to another TO-preplay (n', f', φ') on (A, B) is an injective map $g\colon n \to n'$ such that:

- *$f'(g(i)) = f(i)$ for all $i \in n$,*
- *$\varphi'(g(i)) = g(\varphi(i))$ for all $i \in n$ (with the convention that $g(0) = 0$),*
- *$g(2i) = g(2i - 1) + 1$ for all $i \in n/2$.*

*The last condition states that g should preserve blocks of an Opponent move and the next Proponent move (so-called OP-*blocks*).*

TO-preplays on (A, B) and morphisms of such form a category $\mathbb{PP}_{A,B}$ with the obvious identities and composition.

Definition 2.1.52. *A* TO-play *on the pair of arenas (A, B) is a TO-preplay on (A, B) of even length.*

We denote by $\mathbb{P}_{A,B}$ the category of TO-plays on (A, B), which is the full subcategory of $\mathbb{PP}_{A,B}$ spanning TO-plays.

Definition 2.1.53. *If $s = (n, f, \varphi)$ is a justified sequence, i and j are in n, and $\varphi(j) = 0$, we say that i is* hereditarily justified *by j if $i = j$ or $\varphi(i)$ is hereditarily justified by j.*

A thread *of s is a maximal sub-sequence of s in which all moves have the same hereditary justifier. If s is a play, then all threads of s are justified sequences in which the pointers are inherited from s.*

Definition 2.1.54. *A* TO-preview *on (A, B) is a TO-preplay $s = (n, f, \varphi)$ such that $\lceil s \rceil_n = s$.*

Let $\mathbb{PV}_{A,B}$ be the full subcategory of $\mathbb{PP}_{A,B}$ spanning previews.

Definition 2.1.55. *A* TO-view *on (A, B) is a non-empty TO-preview of even length. Let $\mathbb{V}_{A,B}$ denote the full subcategory of $\mathbb{PV}_{A,B}$ spanning TO-views.*

Remark. *$\mathbb{V}_{A,B}$ is also a full subcategory of $\mathbb{P}_{A,B}$.*

Note that the notions of views and plays are exactly the same as in HON games, but that the notions of morphisms are different.

The embedding $\mathsf{i}_{TO}\colon \mathbb{V}_{A,B} \hookrightarrow \mathbb{P}_{A,B}$ induces an adjunction

$$\begin{tikzcd}
\widehat{\mathbb{P}_{A,B}} \arrow[r, bend left, "\Delta_{\mathsf{i}_{TO}}"] \arrow[r, phantom, "\perp"] & \widehat{\mathbb{V}_{A,B}}, \arrow[l, bend left, "\prod_{\mathsf{i}_{TO}}"]
\end{tikzcd}$$

where Δ_F denotes restriction along F^{op} and $\prod_F$ denotes right Kan extension along F^{op} (see Section 2.2.4).

Definition 2.1.56. *We call* $\widehat{\mathbb{V}_{A,B}}$ *the category of* TO-behaviours *and* $\widehat{\mathbb{P}_{A,B}}$ *that of* TO-strategies*. We denote by* $\mathrm{Sh}(\mathbb{P}_{A,B})$ *the category of* innocent TO-strategies *on* (A,B)*, which is the essential image of* $\prod_{\mathsf{i}_{TO}}$*.*

By Theorem 2.2.84, full faithfulness of i_{TO} entails that $\prod_{\mathsf{i}_{TO}}$ restricts to an equivalence $\mathrm{Sh}(\mathbb{P}_{A,B}) \simeq \widehat{\mathbb{V}_{A,B}}$.

Remark. *As the notation* Sh(−) *suggests, this is also a category of* sheaves *for the Grothendieck topology induced by the embedding* $\mathbb{V}_{A,B}$ *into* $\mathbb{P}_{A,B}$ *(see Section 2.2.7).*

2.1.4 Abramsky-Jagadeesan-Malacaria Games

Required: ∅.
Recommended: 2.1.1.

AJM games are another approach to game semantics, and we once again follow Harmer [46] to define them. These games are slightly different from HON games in the sense that they are not based on arenas, but on even simpler data: moves inside an "arena" are not even ordered, and there are no justification pointers. Because of this, we need a predicate to tell which sequences are valid and which are not.

Definition 2.1.57. *A* game *is a triple* $G = (P, O, \approx)$*, where:*

- *P and O are disjoint sets of moves, with P called the Proponent moves (or simply P-moves) and O the Opponent moves (or simply O-moves). We denote by M the set of all moves $P + O$. A* legal play *is an alternating sequence of moves starting with an O-move. Formally, a legal play s is a pair (n, f) of a natural number n and a function $f : n \to M$ such that, for all i in n, $f(i)$ is in O if and only if i is odd. Legal plays equipped with prefix ordering form a category $\mathbb{L}_G$.*
- *And where $\approx$ is a partial equivalence relation, i.e., a symmetric, transitive (but not necessarily reflexive) relation verifying:*
  - *$\approx$ is length- and prefix-preserving, i.e., if $s \approx t$, then $|s| = |t|$, and if $sa \approx tb$, then $s \approx t$,*
  - *$\approx$ is* extensible*, i.e., if $s \approx t$ and $sa \approx sa$, then there exists b such that $sa \approx tb$.*

  A play *is a legal play s such that $s \approx s$. Plays on G equipped with prefix ordering form a category $\mathbb{P}_G$.*

Not only does $\approx$ define the set of plays, but it also serves another purpose: it can be thought of as a way to equate plays that are basically the same. There are thus two possible choices to define strategies: as a set of plays, or as a set of equivalence classes for $\approx$.

To define plays on pairs of games, we first define arrow games.

Definition 2.1.58. *If G and H are games, then $G \multimap H$ is the game given by:*

- $P_{G \multimap H} = O_G + P_H$,

- $O_{G \multimap H} = P_G + O_H$,
- *and if* $s = (n, f)$ *and* $t = (m, g)$, $s \approx_{G \multimap H} t$ *if and only if:*
  - *for all* i *in* n, $f(i) \in P_{G \multimap H}$ *exactly when* $g(i) \in P_{G \multimap H}$
  - *and* $s_G \approx_G t_G$ *and* $s_H \approx_H t_H$, *where* s_G *is the projection of* s *onto* G, *and similarly for* s_G, t_H, *and* t_H.

We can now define plays on a pair of games (G, H):

Definition 2.1.59. *A* play *on the pair of games* (G, H) *is a play of even length on the game* $G \multimap H$. *Plays on* (G, H) *equipped with prefix ordering form a category* $\mathbb{P}_{G,H}$.

Once the notion of play is defined, strategies are defined just like in the case of HON games:

Definition 2.1.60. *A* strategy *on* (G, H) *is a prefix-closed set of plays on* (G, H).

Like in HON games and their variations, composition of strategies is based on the notion of interaction sequence, which is defined similarly.

Definition 2.1.61. *An* interaction sequence *on the tuple of games* (G, H, I) *is a sequence of elements of* $M_G + M_H + M_I$ *whose projections to* (G, H) *and* (H, I) *are plays. Interaction sequences on* (G, H, I) *equipped with prefix ordering form a category* $\mathbb{P}_{G,H,I}$.

Note that there is a slight difference between this definition and the definitions of interaction sequences in the other cases because we do not require that the last move belong either to G or I.

Composition of strategies is unsurprisingly defined as:

Definition 2.1.62. *If* σ *and* τ *are strategies on* (G, H) *and* (H, I) *respectively, then the* composite $\sigma;\tau$ *of* σ *and* τ *is the strategy on* (G, I) *that accepts a play* s *if and only if there is an interaction sequence* u *on* (G, H, I) *such that:*

- *the projection of* u *to* (G, I) *is* s,
- *and the projections of* u *to* (G, H) *and* (H, I) *are accepted by* σ *and* τ *respectively.*

The proof of associativity of composition of strategies once again relies on a zipping lemma, which is expressed just like 2.1.16, where generalised interaction sequences are defined as:

Definition 2.1.63. *A* generalised interaction sequence *on the tuple of games* (G, H, I, J) *is a sequence of elements in* $M_G + M_H + M_I + M_J$ *whose projections to* (G, H), (H, I), *and* (I, J) *are plays.*

As we said before, since $s \approx t$ means that s and t are essentially the same play, there is another notion of strategy, which is given by equating all plays that belong to the same equivalence class of $\approx$.

Definition 2.1.64. *A* saturated strategy *on the pair of games (G, H) is a prefix-closed set of equivalence classes of $\approx_{G \multimap H}$. Saturated strategies equipped with prefix ordering form a category $\mathbb{P}^s_{G,H}$.*

For this definition to make sense, we should verify that the prefix relation on plays extends to a poset on equivalence classes of $\approx$, which is easily seen to be true.

Saturated strategies accept plays only based on the equivalence class they belong to. They can be thought of as strategies that only look at plays from afar to choose whether to accept them or not, and two plays that are essentially the same must either both be accepted or both rejected.

There is an obvious bijection between saturated strategies and strategies σ such that, if σ accepts s, then it also accepts all $s' \approx s$. Let us call such strategies *replete*. In this light, composition of saturated strategies amounts to showing that the composite $\sigma; \tau$ of two replete strategies σ and τ is again replete, which is easy to check.

2.1.5 Blass Games

Required: $\varnothing$.
Recommended: $\varnothing$.

Blass games [14, 15] are the last games that we will study in this dissertation. They are well-known for the fact that composition of strategies is not associative, and we will use them to test the limits of our framework for game semantics in Chapter 6.

Blass games are different from arena games on several aspects:

- contrary to arena games, where the graph structure describes the possible moves that can occur in a play, the structure of Blass games describes the set of all positions in the game, which is a convention used in many models that do not have justification pointers [3, 55, 47],
- they come equipped with a notion of winning position, so they are closer to game theory,
- plays do not necessarily start by an Opponent move.

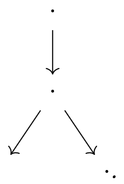
Since we will not be interested in winning strategies, but only in how to compose strategies, we will skip the description of winning positions for a simpler presentation.

Definition 2.1.65. *A game G is a pair (P, s) where P is a simple tree, i.e., a directed, simple graph in which each vertex is uniquely reachable from a fixed vertex without parent, called the* root*; and $s \in \{P, O\}$.*

Terminology 2.1.66. *In a game G, P is called the set of* positions *and s is called the* starting player*. Each position in the game G is given a polarity that is either P or O, depending on its depth in P: if the position is a root, then its polarity is s, otherwise it is the opposite of its parent's polarity. A position is an O-position if its polarity is O, and a P-position otherwise.*

We think of O-positions as positions where it is O's turn to play, and P-positions as those where it is P's turn to play.

Example 2.1.67. *The game tree for the boolean game is the pair* $(\mathbb{B}, O)$*, where* $\mathbb{B}$ *is the tree*



In the initial position, the program and the environment have not interacted yet; in the second one, the environment has asked the program for its return value; and in the bottom positions, the program has answered a value, either `true` *or* `false`*, depending on the position.*

We now want to build arrow games to define strategies on a pair of games, which are our candidate morphisms of a category of Blass games and strategies. To make this definition more tractable, we adopt notations inspired from Abramsky [2].

Definition 2.1.68. *Games are defined coinductively by the grammar:*

$$G ::= \coprod_{i \in I} G \mid \prod_{i \in I} G.$$

This definition is equivalent to the previous one, in which we drop the constraint that both players should alternate. These games are basically what Blass calls *relaxed games*, except that they do not verify the constraint that a player should only play finitely many times in a row (but Blass writes that this constraint exists only for technical reasons). The translation from the "grammatical form" to the "tree form" goes as follows:

- if G_i translates to a game (T_i, s_i), then $\coprod_{i \in I} G_i$ translates to the game (T, P), where T is the tree with I edges from its root, whose ith sub-tree is T_i, and in which, after P plays i, it is s_i's turn to play,
- similarly for $\prod_{i \in I} G_i$, except the initial player in the translation is O.

Games of the form $\coprod_{i \in I} G_i$ are thus "positive" games in which P plays first, and games of the form $\prod_{i \in I} G_i$ are negative ones, in which O plays first.

Example 2.1.69. *The empty product* $\prod_{i \in 0} G_i$ *is the game* $(\cdot, O)$*, i.e., the game whose only position is an O-position. Similarly, the empty coproduct* $\coprod_{i \in 0} G_i$ *is the game* $(\cdot, P)$*, i.e., the game whose only position is a P-position. Since these games do not depend on the* G_i*'s used to define them, we will write them* $\prod_0$ *and* $\coprod_0$ *for the sake of brevity.*

Based on this notation, all finite games can be written as a product $\prod_n G_n$ *or a coproduct* $\coprod_n G_n$*, where* G_n *are finite games. For example, the boolean game defined in Example 2.1.67 can be written as* $\prod_1 \coprod_2 \prod_0$*. Indeed, the top position is an O-position, so it is a product, and it has one child, so it is of the* *form $\prod_1 G$. Similarly, G's top position is a P-position, so it is a coproduct, and it has two children, so it is of the form $H_1 + H_2$. Finally, H_1 and H_2 are both $\prod_0$, whence the final expression.*

We may now define arrow games:

Definition 2.1.70. *If G and H are games, then the arrow game $G \multimap H$ is defined coinductively as:*

- $\coprod_{i\in I} \left(G_i \multimap \left(\coprod_{j\in J} H_j\right)\right) + \coprod_{j\in J} \left(\left(\prod_{i\in I} G_i\right) \multimap H_j\right)$ *if G is of the form $\prod_{i\in I} G_i$ and H of the form $\coprod_{j\in J} H_j$, where $+$ is a binary $\coprod$ (and the whole expression should be understood as a $\coprod_{i\in I+J}$ of games, rather than a $\coprod_{i\in 2}$ of coproducts of games),*
- $\prod_{i\in I} \left(G_i \multimap H\right)$ *if G is of the form $\coprod_{i\in I} G_i$ and H of the form $\coprod_{j\in J} H_j$,*
- $\prod_{j\in J} \left(G \multimap H_j\right)$ *if G is of the form $\prod_{i\in I} G_i$ and H of the form $\prod_{j\in J} H_j$,*
- $\prod_{(i,j)\in I\times J} \left(G_i \multimap H_j\right)$ *otherwise.*

The intuition behind this definition is that it defines an interleaving of all moves of G and H, but imposing the condition that Opponent should play whenever they can. For example, if G and H are both coproducts, then they are both positive games in which P plays first, but since polarity is reversed in the left-hand game (like in arena games), O may play in that game, so they should, hence the form of the result: it is a product because Opponent plays first, and after they have played, the play may continue in either the right-hand game or one of the sub-games of the left-hand one. The idea is similar when G and H are both products: O may play in the right-hand game, so they do. When G is a product and H is a coproduct, O cannot play in any of the two games, so P chooses one of them to play in and a move in it. Following this logic, when G is a coproduct and H a product, O should pick one of the two games to play in and a move in it, and we should get a similar expression for $G \multimap H$. However, this would break alternation, by allowing O to play twice in a row, even if G and H alternate. To solve this problem, we let O play in both G and H in a single move, whence the expression of $G \multimap H$.

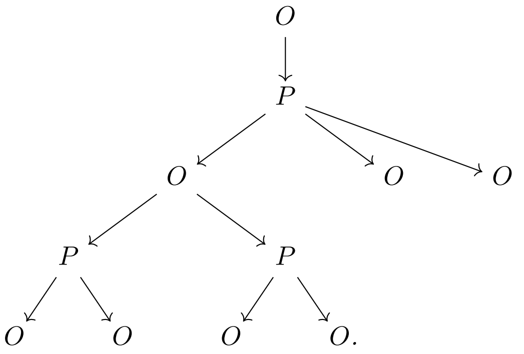
This may also be defined by seeing $G \multimap H$ as the forest consisting of G and H's game trees. The rule to decide which player's turn it is to play in $G \multimap H$ is that Opponent always plays first. To make a move in $G \multimap H$:

- if it is Proponent's turn to play, it means that it is their turn to play both in G and H, so they can choose in which game to play and make a move there,
- if it is Opponent's turn to play in G or H (but not both), then they play in that game,
- otherwise, for the same reason as above, Opponent plays in both G and H at the same time.

Example 2.1.71. *The game $\mathbb{B} \multimap \mathbb{B}$ can be computed as:*

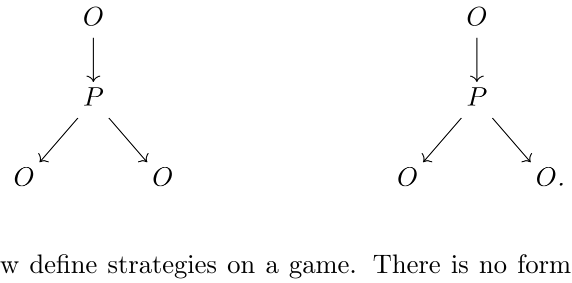
$$\begin{aligned}
\mathbb{B} \multimap \mathbb{B} &= (\textstyle\prod_1 \coprod_2 \prod_0) \multimap (\prod_1 \coprod_2 \prod_0)\\
&= \textstyle\prod_1((\prod_1 \coprod_2 \prod_0) \multimap (\coprod_2 \prod_0))\\
&= \textstyle\prod_1((\coprod_1((\coprod_2 \prod_0) \multimap (\coprod_2 \prod_0))) + (\coprod_2((\prod_1 \coprod_2 \prod_0) \multimap \prod_0)))\\
&= \textstyle\prod_1((\coprod_1(\prod_2(\prod_0 \multimap (\coprod_2 \prod_0)))) + (\coprod_2 \prod_0))\\
&= \textstyle\prod_1((\coprod_1(\prod_2(\coprod_0 + \coprod_2(\prod_0 \multimap \prod_0)))) + (\coprod_2 \prod_0))\\
&= \textstyle\prod_1((\coprod_1 \prod_2 \coprod_2 \prod_0) + (\coprod_2 \prod_0)),
\end{aligned}$$

which, when drawn as a tree whose nodes are labelled O or P based on their polarities (this is necessary since G and H may not be alternating, though they are in this case), gives



The first move in $\mathbb{B}_l \multimap \mathbb{B}_r$ (where, once again, subscripts only serve to distinguish between the two copies of $\mathbb{B}$) corresponds to O playing its initial move in $\mathbb{B}_r$. Its two right-hand-side children correspond to P answering directly in $\mathbb{B}$, thus ending the game: it is O's turn to play in $\mathbb{B}_r$, and therefore also in $\mathbb{B}_l \multimap \mathbb{B}_r$, but there are no moves to play. Its left-hand child, however, corresponds to P playing the initial move in $\mathbb{B}_l$, which is possible because polarities are reversed. Its two children correspond to the two possible moves O can answer in $\mathbb{B}_l$, and the subsequent children to the different choices P can make in $\mathbb{B}_r$.

Notice how this is actually just describing the different interleavings of moves of $\mathbb{B}_l$ and $\mathbb{B}_r$ under the constraint that Opponent always plays first. We can thus also represent it as the product of $\mathbb{B}_l$ and $\mathbb{B}_r$, remembering that polarity is actually reversed in $\mathbb{B}_l$, and the fact that Opponent plays as soon as possible:



We may now define strategies on a game. There is no formal definition of strategies on Blass games, so their definition is open to interpretation. The following definition is our own interpretation of strategies on Blass games, and we do not claim that they are the same strategies as the ones Blass had in mind. In particular, our definition definitely clashes with the one given by Abramsky, who only considers total strategies. Nevertheless, we believe that the reason why strategies fail to compose in Blass games is left unchanged with our definition of strategy.

Definition 2.1.72. *The set $S(G)$ of* strategies *on the game G is defined coinductively as:*

- $S(\coprod_{i\in\varnothing} G_i) = \{\star\}$,
- $S(\coprod_{i\in I} G_i) = \sum_{i\in I} S(G_i) + \{\star\}$ *otherwise,*
- $S(\prod_{i\in I} G_i) = \prod_{i\in I} S(G_i)$.

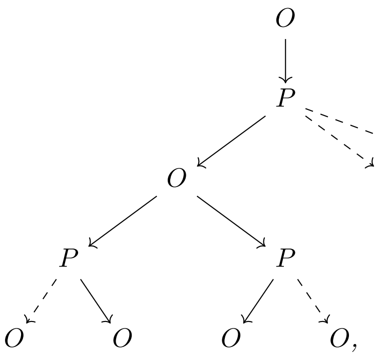
In other words, a strategy σ on G is a downwards-closed set of positions of G (or upwards-closed, if we imagine the tree's root is at the top) such that:

- for all P-positions p in σ, σ contains at most one of p's children,
- for all O-positions p in σ, σ contains all of p's children.

This indeed corresponds with the intuition of a strategy (or at least of a deterministic strategy): Proponent chooses which move to make for each position in which they have a choice to make. It may also be viewed as a partial function from P-positions of G to their children (that is however slightly wrong because two functions that differ only on a part of G that can never be reached by that strategy should be considered equal).

A strategy on a pair of games (G, H) is a strategy on the game $G \multimap H$. By the description of $G \multimap H$ given above, a strategy on $G \multimap H$ is basically a function from pairs (p, q) of an O-position p of G and a P-position q of H to the union of p and q's children.

Example 2.1.73. *Let us describe the strategy associated to the* not *function on booleans. It must be the sub-tree of the tree drawn in Example 2.1.71 that corresponds to the function asking for its argument and then answering the negation of that argument. It may be drawn as*



where the dashed arrows are the part of the tree that have been cut off.

If we call ε, q, t, and f the positions of $\mathbb{B}$, we can also see this strategy on the pair $(\mathbb{B}_l, \mathbb{B}_r)$ as the function: $(\varepsilon_l, q_r) \mapsto q_l$, $(\mathsf{t}_l, q_r) \mapsto \mathsf{f}_r$, $(\mathsf{f}_l, q_r) \mapsto \mathsf{t}_r$.

We will not define composition of strategies formally, but only give an idea of how it works. To the best of our knowledge, the formal description of composition of strategies in Blass games is not written anywhere (at least not in [14, 15, 2]). Composition of strategies is explained as follows in [14].

Given a strategy σ on (G, H) and a strategy τ on (H, I), the composite $\sigma;\tau$ on (G, I) plays as if, along with G and I's game trees, H's game tree were also present. Since σ is basically a rule that explains how P should play in B and O in A, and τ how P should play in C and O in B, it is not very surprising that this data can be used to create a rule for P to play in C and O in A. It plays according to either σ or τ, depending on the polarities of the positions it has reached in A, C, and the fictitious copy of B. All the possible cases are in the table below.

G	H	I	σ	τ	$\sigma;\tau$	what to do
O	O	O	H	I	I	I
O	O	P	H	$H \mid I$	$G \mid I$	τ
O	P	O	$G \mid H$	$H \,\&\, I$	I	$\sigma \leftrightsquigarrow (H, I)$
O	P	P	$G \mid H$	H	$G \mid I$	σ
P	O	O	$G \,\&\, H$	I	$G \,\&\, I$	$\tau \rightsquigarrow \sigma$
P	O	P	$G \,\&\, H$	$H \mid I$	G	$(G, H) \leftrightsquigarrow \tau$
P	P	O	G	$H \,\&\, I$	$G \,\&\, I$	$\sigma \rightsquigarrow \tau$
P	P	P	G	H	G	G

The table is a rule to know what move to play depending on the polarities of the current positions in G, H, and I, and is meant to be read as follows: the first three columns enumerate all the possible combinations of polarities in G, H, and I; the next two columns show where σ and τ will play their next moves; the next column shows where $\sigma;\tau$ should make its next move; and the final column gives a "formula" to compute that move:

- if the formula is a game, then we are in an O position in which Opponent must play exactly in that game, so we wait for them to do so,

- if the formula is a strategy, then we are in a P position, and we play according to the strategy, leaving the other games untouched, which may change the "internal state" of the strategy (the position on H),

- if the formula is $\sigma \leftrightsquigarrow (H, I)$ (or the symmetric one for τ), then we are in an O position in which Opponent must play in exactly one of the external games, so we wait for Opponent to make their move (m_H, m_I) on $H \multimap I$, and then check that σ indeed accepts to play m_H: if σ does not accept to play m_H, there is some sort of conflict between the environment and the strategy, and the composite strategy stops playing as soon as it is Proponent's turn to play,

- if the formula is $\tau \rightsquigarrow \sigma$ (or the symmetric one), then we are in an O position where Opponent must play in both games, so we first let Opponent play m_I on I: if the final position of m_I is an O position, there is a conflict and the strategy stops playing as soon as it is Proponent's turn, otherwise we then play like in $\sigma \leftrightsquigarrow (H, I)$ (this whole process should be atomic, in the sense that Opponent will actually play (m_G, m_I) in the game, rather than m_I followed by m_G).

Example 2.1.74. *Let us show how to compose the strategy associated to the* `not` *function with itself. Let us call ε, q, t, and f the four positions of $\mathbb{B}$ and σ the strategy for* `not`*.*

At the very beginning, the position is $(\varepsilon, \varepsilon, \varepsilon)$, which has polarity (O, O, O), so we simply wait for Opponent to move in the right-hand copy of $\mathbb{B}$.

Since the only move they can make is q, we are then in $(\varepsilon, \varepsilon, q)$, which has polarity (O, O, P), so we play according to σ on the right-hand copies of $\mathbb{B}$, which are in position (ε, q), so we should play q in the middle copy of $\mathbb{B}$.

Describing the whole strategy would take some time, but the result is exactly as expected: it maps (ε_l, q_r) to q_l (with internal state q_m), (t_l, q_r) to t_r (with internal state f_m), and (f_l, q_r) to f_r (with internal state t_m).

We are now ready to exhibit an example where composition of strategies is not associative.

Example 2.1.75. *Take four games $G = \prod_1 \coprod_0$, $H = J = \coprod_0$, and $I = \prod_0$. In terms of trees, G has only one branch of length 1, while all the other games are trivial. Both G and I have polarity O, while H and J have polarity P. If we call $\star$ the non-initial position of G, we define σ on $G \multimap H$ as the strategy that maps $(\varepsilon_G, \varepsilon_H)$ to $\star$. Since $H \multimap I$ is trivial, there is only one strategy τ on it, and similarly, since $I \multimap J$ is trivial, there is only one strategy υ on it.*

The composite $\sigma;\tau$ starts on $(\varepsilon_G, \varepsilon_H, \varepsilon_I)$, which has polarity (O, P, O), the strategy thus waits for Opponent input on I, but it never gets any because this game is trivial, so the strategy does not do anything (note that $G \multimap I$ is trivial, so the strategy could not have played anything anyway). The composite $(\sigma;\tau);\upsilon$ starts on $(\varepsilon_G, \varepsilon_I, \varepsilon_J)$, which has polarity (O, O, P), so the strategy plays according to υ, which does nothing.

On the other hand, $\tau;\upsilon$ starts on $(\varepsilon_H, \varepsilon_I, \varepsilon_J)$, which has polarity (P, O, P), so the strategy waits for Opponent input on I, which never comes, so nothing happens. But then, $\sigma;(\tau;\upsilon)$ starts on $(\varepsilon_G, \varepsilon_H, \varepsilon_J)$, which has polarity (O, P, P), so it plays according to σ, which plays $\star$ on A.

We thus have that $\sigma;(\tau;\upsilon) \neq (\sigma;\tau);\upsilon$, so composition of strategies is not associative.

2.2 Categorical Preliminaries

We assume basic knowledge of category theory, such as the notions of categories, functors, natural transformations, limits and colimits, and adjunctions. We refer the reader to the classical reference [76] for definitions of these notions.

We write composition of maps in the same order as composition of functions, as is usual in category theory, so $g \circ f$ is the composite $A \xrightarrow{f} B \xrightarrow{g} C$. We will also often write gf instead of $g \circ f$.

Given a category $\mathbb{C}$ and objects c and c' of $\mathbb{C}$, we will usually denote by $[c, c']$ the homset $\mathbb{C}(c, c')$ and let the context disambiguate.

2.2.1 Comma and Cocomma Categories

Required: $\varnothing$.
Recommended: $\varnothing$.

Comma [76] and cocomma categories [89] will be useful throughout this dissertation, mainly for their combinatorial flavour, which will prove useful to build categories.

Definition 2.2.1 (Comma Category)**.** *Let* $\mathbb{A} \xrightarrow{F} \mathbb{C} \xleftarrow{G} \mathbb{B}$ *be a cospan of categories, the comma category* $F \downarrow G$ *is the universal cone equipped with a natural transformation* λ *as below left. In other words, it is the pair of a cone and a natural transformation* λ *such that, for all other cones and natural transformations* α*, there is a unique mediating arrow* $\langle U, V\rangle$ *as below right such that* $\partial\langle U, V\rangle = U$*,* $б\langle U, V\rangle = V$*, and* $\lambda \cdot \langle U, V\rangle = \alpha$*.*

$$\begin{tikzcd}
F \downarrow G \arrow[r, "б"] \arrow[d, "\partial"'] & \mathbb{B} \arrow[d, "G"] \\
\mathbb{A} \arrow[r, "F"'] \arrow[ur, phantom, "\overset{\lambda}{\Longrightarrow}"] & \mathbb{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
X \arrow[dr, dashed, "{[U,V]}"] \arrow[drr, bend left, "V"] \arrow[ddr, bend right, "U"'] & & \\
& F \downarrow G \arrow[r, "б"] \arrow[d, "\partial"'] & \mathbb{B} \arrow[d, "G"] \\
& \mathbb{A} \arrow[r, "F"'] \arrow[ur, phantom, "\overset{\lambda}{\Longrightarrow}"] & \mathbb{C}
\end{tikzcd}$$

If we unfold the definitions, we get:

- the objects of $F \downarrow G$ are triples (a, b, f) of two objects a of $\mathbb{A}$, b of $\mathbb{B}$, and a morphism $f : Fa \to Gb$,
- the morphisms from (a, b, f) to (a', b', f') are pairs of morphisms (φ, ψ) such that

$$\begin{tikzcd}
Fa \arrow[r, "F\varphi"] \arrow[d, "f"'] & Fa' \arrow[d, "g"] \\
Gb \arrow[r, "G\psi"'] & Gb'
\end{tikzcd}$$

  commutes and the identities and composition are defined in the obvious way,
- $\partial : F \downarrow G \to \mathbb{A}$ maps (a, b, f) to a and (φ, ψ) to φ,
- $б : F \downarrow G \to \mathbb{B}$ maps (a, b, f) to b and (φ, ψ) to ψ.

Dually, there is a notion of cocomma category:

Definition 2.2.2 (Cocomma Category)**.** *Let* $\mathbb{A} \xleftarrow{F} \mathbb{C} \xrightarrow{G} \mathbb{B}$ *be a span of categories, the cocomma category* $F \uparrow G$ *is the universal cocone equipped with a natural transformation* λ *as below left. In other words, it is the pair of a cocone and a natural transformation* λ *such that, for all other cocones and natural transformations* α*, there is a unique mediating arrow* $[U, V]$ *as below right such that* $[U, V]l = U$*,* $[U, V]r = V$*, and* $[U, V] \cdot \lambda = \alpha$*.*

$$\begin{tikzcd}
\mathbb{C} \arrow[r, "G"] \arrow[d, "F"'] & \mathbb{B} \arrow[d, "r"] \\
\mathbb{A} \arrow[r, "l"'] \arrow[ur, phantom, "\overset{\lambda}{\Longrightarrow}"] & F \uparrow G
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbb{C} \arrow[r, "G"] \arrow[d, "F"'] & \mathbb{B} \arrow[d, "r"] \arrow[ddr, bend left, "V"] & \\
\mathbb{A} \arrow[r, "l"'] \arrow[ur, phantom, "\overset{\lambda}{\Longrightarrow}"] \arrow[drr, bend right, "U"'] & F \uparrow G \arrow[dr, dashed, "{\langle U,V \rangle}"'] & \\
& & X
\end{tikzcd}$$

The explicit description is slightly more complex than that of comma squares:

- objects of $F \uparrow G$ are the disjoint union of those of $\mathbb{A}$ and $\mathbb{B}$,
- for morphisms:
  - $(F \uparrow G)(a, a') = \mathbb{A}(a, a')$,
  - $(F \uparrow G)(b, b') = \mathbb{B}(b, b')$,
  - $(F \uparrow G)(a, b) = \sum_{c \in \mathbb{C}} \mathbb{A}(a, Fc) \times \mathbb{B}(Gc, b)$, where such triples are quotiented by the following equivalence relation: two triples (c, f, g) and (c', f', g') are equivalent when there exists a zigzag $c \xrightarrow{f_0}\xleftarrow{f_1} \dots \xrightarrow{f_{n-1}}\xleftarrow{f_n} c'$ and "lanterns", i.e., commuting diagrams of the form

$$\begin{tikzcd}
 & & & a \arrow[dlll, "f"'] \arrow[dll] \arrow[dr] \arrow[drr, "f'"] & & \\
Fc \arrow[r, "Ff_0"'] & \cdot & \dots \arrow[l, "Ff_1"'] \arrow[rr, "Ff_{n-1}"'] & & \cdot & Fc' \arrow[l, "Ff_n"']
\end{tikzcd}
\qquad
\begin{tikzcd}
Gc \arrow[r, "Gf_0"] \arrow[drr, "g"'] & \cdot \arrow[dr] & \dots \arrow[l, "Gf_1"'] \arrow[r, "Gf_{n-1}"] & \cdot \arrow[dl] & Gc' \arrow[l, "Gf_n"'] \arrow[dll, "g'"] \\
 & & b, & &
\end{tikzcd}$$

  - $(F \uparrow G)(b, a) = \varnothing$,

  and the identities and composition are defined in the obvious way,
- $l: \mathbb{A} \to F \uparrow G$ maps $\mathbb{A}$ to its copy in $F \uparrow G$,
- $r: \mathbb{B} \to F \uparrow G$ maps $\mathbb{B}$ to its copy in $F \uparrow G$.

2.2.2 Fibrations

Required: $\varnothing$.
Recommended: $\varnothing$.

In this dissertation, we will use two notions of fibrations: one that is simply called *fibration* and the other that is called *discrete fibration*. Basically, a functor $p: \mathbb{E} \to \mathbb{B}$ is a fibration when there is a canonical way to find antecedents of morphisms $b \to p(e)$.

Definition 2.2.3. *For any functor $p: \mathbb{E} \to \mathbb{B}$, a morphism $r: e' \to e$ in $\mathbb{E}$ is* cartesian *when, as below, for all $t: e'' \to e$ and $k: p(e'') \to p(e')$ such that $p(r) \circ k = p(t)$, there exists a unique $s: e'' \to e'$ such that $p(s) = k$ and $r \circ s = t$:*

$$\begin{tikzcd}
e'' \arrow[dr, dashed, "s"'] \arrow[drr, bend left, "t"] \arrow[dd] & & \\
 & e' \arrow[r, "r"'] \arrow[dd] & e \arrow[dd] \\
p(e'') \arrow[dr, "k"'] \arrow[drr, bend left, "p(t)"] & & \\
 & p(e') \arrow[r, "p(r)"'] & p(e).
\end{tikzcd}$$

Definition 2.2.4 (Fibration)**.** *A functor $p: \mathbb{E} \to \mathbb{B}$ is a* fibration *if and only if for all e in $\mathbb{E}$, any $h: b' \to p(e)$ has a cartesian lifting, i.e., a cartesian antecedent by p with codomain e.*

Let us now describe discrete fibrations:

Definition 2.2.5 (Discrete Fibration)**.** *A functor $p{:}\,\mathbb{E} \to \mathbb{B}$ is a* discrete fibration *when, for all e in $\mathbb{E}$ and morphisms $f{:}\,b \to p(e)$, there exists a unique morphism $g{:}\,e' \to e$ such that $p(g) = f$. Such a g is called a* cartesian lifting *of e along f.*

A morphism of discrete fibrations $p{:}\,\mathbb{E} \to \mathbb{B}$ and $q{:}\,\mathbb{E}' \to \mathbb{B}$ is a functor $r{:}\,\mathbb{E} \to \mathbb{E}'$ such that $qr = p$.

Discrete fibrations over $\mathbb{B}$ and morphisms of such form a category $\mathsf{DFib}_{\mathbb{B}}$*.*

Note that a discrete fibration is basically a fibration whose fibres are discrete, in the sense that there is a unique morphism above an $f{:}\,b \to p(e)$ whose codomain is e.

Notation 2.2.6. *Since the cartesian lifting of e along f is unique, we may denote it by $e \uparrow f$, and we denote the domain of this map by $e \cdot f$.*

It may be useful to draw a picture here. The property of discrete fibration states that, if p is a discrete fibration, then for any solid part of the drawing below, there is a unique dashed part that completes the drawing.

$$\begin{tikzcd}
x' \arrow[rr, dashed, "u"{name=U}] \arrow[d, dashed, mapsto, "p"'] && x \arrow[d, mapsto, "p"] \\
a' \arrow[rr, "f"'{name=F}] && a
\arrow[from=U, to=F, dashed, mapsto, "p"]
\end{tikzcd}$$

Notice that, by uniqueness, it suffices to find a u as above to prove that $x \uparrow f = u$.

Remark. *Since we will sometimes mix cartesian liftings of different discrete fibrations and actions of morphisms in different presheaves in the proofs, we decide to write the action of morphisms in a presheaf X as $x \cdot_X f$ and the cartesian lifting of e along f for the discrete fibration p as $e \uparrow_p f$, and $e \cdot_p f$ for its domain when there may be an ambiguity.*

Note that this clash of notations happens for a good reason (if there is such a thing as a good reason for a clash of notations): we will show in Section 2.2.5 that action of morphisms in presheaves correspond in some sense to cartesian lifting in discrete fibrations.

Proposition 2.2.7 ("Functoriality" of cartesian lifting)**.** *For all discrete fibrations $p{:}\,\mathbb{X} \to \mathbb{A}$, cartesian lifting is "functorial", in the sense that, for all object x of $\mathbb{X}$:*

- *for all $f{:}\,a \to p(x)$ and $g{:}\,a' \to a$, $(x \uparrow_p f) \circ ((x \cdot_p f) \uparrow_p g) = x \uparrow_p (fg)$,*
- *$x \uparrow_p id_{p(x)} = id_x$.*

Proof. Both proofs are by uniqueness of cartesian lifting.

- For composition, we have:

$$\begin{tikzcd}
(x \cdot_p f) \cdot_p g \arrow[rr, "(x \cdot_p f) \uparrow_p g"{name=A}] \arrow[d, mapsto, "p"'] && x \cdot_p f \arrow[rr, dash, "x \uparrow_p f"{name=B}] \arrow[d, mapsto, "p"] && x \arrow[d, mapsto, "p"] \\
a' \arrow[rr, "g"'{name=G}] && a \arrow[rr, dash, "f"'{name=F}] && p(x),
\arrow[from=A, to=G, mapsto, "p"]
\arrow[from=B, to=F, mapsto, "p"]
\end{tikzcd}$$

so the morphism satisfies the property of $x \uparrow_p (fg)$, and so they are equal.

- For the identity, we have:

$$\begin{tikzcd}
x \arrow[rr, "id_x"] \arrow[d, mapsto, "p"'] & {} \arrow[d, mapsto, "p"] & x \arrow[d, mapsto, "p"] \\
p(x) \arrow[rr, "id_{p(x)}"'] & {} & p(x),
\end{tikzcd}$$

  so the morphism satisfies the property of $x \uparrow_p id_{p(x)}$, and so they are equal. □

Unsurprisingly, cartesian lifting is compatible with morphisms of discrete fibrations:

Proposition 2.2.8. *If $p{:}\,\mathbb{X} \to \mathbb{A}$ and $q{:}\,\mathbb{Y} \to \mathbb{A}$ are discrete fibrations and $f{:}\,p \to q$ is a morphism of discrete fibrations, then for all objects x of $\mathbb{X}$ and morphisms $g{:}\,a \to p(x)$, $f(x \uparrow_p g) = f(x) \uparrow_q g$.*

Proof. Because $qf = p$, we have:

$$\begin{tikzcd}
x \cdot_p g \arrow[rr, "x \uparrow_p g"] \arrow[d, mapsto, "f"'] & {} \arrow[d, mapsto, "f"] & x \arrow[d, mapsto, "f"] \\
f(x \cdot_p g) \arrow[rr, "f(x \uparrow_p g)"'] \arrow[d, mapsto, "q"'] & {} \arrow[d, mapsto, "q"] & f(x) \arrow[d, mapsto, "q"] \\
a \arrow[rr, dash, "g"'] & {} & p(x),
\end{tikzcd}$$

so by looking only at the bottom square, we see that $f(x \uparrow_p g)$ has the property of $x \uparrow_q g$, so they are equal. □

Finally, let us mention that there is a dual notion to that of discrete fibration, which we will also use at some point:

Definition 2.2.9 (Discrete opfibration)**.** *A functor $p{:}\,\mathbb{E} \to \mathbb{B}$ is a* discrete opfibration *when, for all e in $\mathbb{E}$ and morphisms $f{:}\,p(e) \to b$, there exists a unique morphism $g{:}\,e \to e'$ such that $p(g) = f$. This g is called the* cartesian oplifting *of e along f and is denoted by $f \uparrow e$, and its codomain is denoted by $f \cdot e$.*

Discrete opfibrations enjoy the same kind of properties as discrete fibrations, including all the properties dual to the ones listed above: a simple proof of all these facts is that p is a discrete opfibration if and only if p^{op} is a discrete fibration.

2.2.3 Ends and Coends

Required: ∅.
Recommended: beginning of 2.2.5.

Note that, in this section and the next one, some fundamental examples will use presheaves, so it may be wise to first read the beginning of Section 2.2.5 (up to Definition 2.2.38) to at least know what presheaves are.

Ends and coends are a kind of universal construction, much akin to limits and colimits. They are defined in pretty much the same way:

- limits are terminal objects in the category of cones, and ends are terminal objects in the category of wedges,
- dually, colimits are initial objects in the category of cocones, and coends are initial objects in the category of cowedges.

We thus need to first define wedges:

Definition 2.2.10. *Let $\mathbb{C}$ and $\mathbb{D}$ be categories, and $F: \mathbb{C}^{op} \times \mathbb{C} \to \mathbb{D}$ be a functor (we also say that F is a bifunctor from $\mathbb{C}$ to $\mathbb{D}$). A* wedge *to F is a pair $(w, (e_c))$ of an object w of $\mathbb{D}$ and, for each object c of $\mathbb{C}$, of a map $e_c: w \to F(c,c)$ such that, for all $f: c \to c'$,*

$$\begin{tikzcd}
w \arrow[r, "e_c"] \arrow[d, "e_{c'}"'] & F(c,c) \arrow[d, "F(c,f)"] \\
F(c',c') \arrow[r, "F(f,c)"'] & F(c,c')
\end{tikzcd}$$

commutes.

A wedge (w, e) to F will be denoted $e: w \to F$.

To tell which wedges are terminal, we need to define a notion of morphism between wedges:

Definition 2.2.11. *A morphism between two wedges $e: w \to F$ and $e': w' \to F$ is a morphism $f: w \to w'$ in $\mathbb{D}$ such that, for all c in $\mathbb{C}$, $e_c = e'_c f$.*

Wedges to F and morphisms of such form a category. We may thus state the definition of end:

Definition 2.2.12. *An* end *of F is a terminal object in the category of wedges to F.*

Because terminal objects are unique up to isomorphism, we will sometimes say *the* end of F, instead of *an* end of F. We will also sometimes call w the end of F, when the actual end is the pair (w, e). Of course, categories of wedges need not have terminal objects, so F may not have an end.

Notation 2.2.13. *Let F be a bifunctor from $\mathbb{C}$ to $\mathbb{D}$. If F has an end (w, e), then we write $\int_{c \in \mathbb{C}} F(c,c)$ or $\int_c F(c,c)$ for w, and π_c or p_c for e_c.*

Example 2.2.14. *If F is a bifunctor from $\mathbb{C}$ to Set, then $\int_c F(c,c)$ is a subset of the product of all $F(c,c)$'s. More precisely, it is the subset of the tuples $(x_c)_{c \in \mathbb{C}}$ such that, for all $f: c \to c'$, $F(c,f)(x_c) = F(f,c')(x_{c'})$. In other words, it is the biggest subset X of $\prod_{c \in \mathbb{C}} F(c,c)$ such that*

$$\begin{tikzcd}
X \arrow[r, "\pi_c"] \arrow[d, "\pi_{c'}"'] & F(c,c) \arrow[d, "F(c,f)"] \\
F(c',c') \arrow[r, "F(f,c')"'] & F(c,c')
\end{tikzcd}$$

commutes for all $f: c \to c'$ in $\mathbb{C}$.

Let $F, G: \mathbb{C} \to \mathbb{D}$ be functors between small categories and $[d, d']$ the homset between two objects d and d' of $\mathbb{D}$. If we define $H: \mathbb{C}^{op} \times \mathbb{C} \to \mathsf{Set}$ by $H(c,c') =$

$[F(c), G(c')]$, then $\int_c H(c,c)$ is the set of natural transformations from F to G. Indeed, $\prod_{c\in\mathbb{C}} H(c,c)$ is the set of all transformations (natural or not) between F and G, and the diagram below left commutes when applied to the transformation (λ_c) exactly when the one below right does for f, which is exactly the naturality condition.

$$\begin{tikzcd}
\prod_c H(c,c) \arrow[r, "\pi_c"] \arrow[d, "\pi_{c'}"'] & H(c,c) \arrow[d, "{H(c,f)}"] \\
H(c',c') \arrow[r, "{H(f,c)}"'] & H(c,c')
\end{tikzcd}
\qquad
\begin{tikzcd}
F(c) \arrow[r, "F(f)"] \arrow[d, "\lambda_c"'] & F(c') \arrow[d, "\lambda_{c'}"] \\
G(c) \arrow[r, "G(f)"'] & G(c')
\end{tikzcd}$$

If $\mathbb{P}$ is a preorder viewed as a category and F is a bifunctor from $\mathbb{C}$ to $\mathbb{P}$, then F's end can be computed as a meet. Indeed, F's end is an object $\int_c F(c,c)$ of $\mathbb{P}$ equipped with maps to all $F(c,c)$'s, so it is a lower bound of the set of all $F(c,c)$'s. Furthermore, suppose that X is also a lower bound of the set of all $F(c,c)$'s, then it comes equipped with morphisms $f_c: X \to F(c,c)$ for all objects c of $\mathbb{C}$. But then

$$\begin{tikzcd}
X \arrow[r, "f_c"] \arrow[d, "f_{c'}"'] & F(c,c) \arrow[d, "{F(c,f)}"] \\
F(c',c') \arrow[r, "{F(f,c')}"'] & F(c,c')
\end{tikzcd}$$

necessarily commutes for all $f: c \to c'$, since there is at most one morphism from X to $F(c,c')$. Therefore, by universal property of $\int_c F(c,c)$, there is a morphism $X \to \int_c F(c,c)$, and thus $\int_c F(c,c) = \bigwedge_{c\in\mathbb{C}} F(c,c)$.

Just like colimits are the dual to limits, ends also have a dual notion, which is (unsurprisingly) called *coends*.

Definition 2.2.15. *If $F: \mathbb{C}^{op} \times \mathbb{C} \to \mathbb{D}$ is a functor, a* cowedge *from F is a pair $(w, (e_c))$ of an object w of $\mathbb{D}$ and, for all objects c of $\mathbb{C}$, a morphism $e_c: F(c,c) \to w$ in $\mathbb{D}$ such that, for all $f: c \to c'$,*

$$\begin{tikzcd}
F(c',c) \arrow[r, "{F(f,c)}"] \arrow[d, "{F(c',f)}"'] & F(c,c) \arrow[d, "e_c"] \\
F(c',c') \arrow[r, "e_{c'}"'] & w
\end{tikzcd}$$

commutes.

A cowedge (w,e) will be denoted by $e: F \to w$. A morphism from a cowedge (w,e) to (w',e') is a morphism $f: w \to w'$ such that, for all objects c of $\mathbb{C}$, $e'_c = fe_c$. Cowedges from F and morphisms of such form a category.

A coend *of F is an initial object in the category of cowedges from F.*

Notation 2.2.16. *Let $F: \mathbb{C}^{op} \times \mathbb{C} \to \mathbb{D}$ be a functor. If F has a coend (w,e), then we write $\int^{c\in\mathbb{C}} F(c,c)$ or $\int^c F(c,c)$ for w_c, and i_c for e_c.*

Example 2.2.17. *If F is a bifunctor from $\mathbb{C}$ to* Set*, its coend $\int^c F(c,c)$ is a quotient of the coproduct $\sum_{c\in\mathbb{C}} F(c,c)$. More precisely, it is the coproduct $\sum_{c\in\mathbb{C}} F(c,c)$ quotiented by the equivalence relation generated by equating an element x in $F(c,c)$ and an element x' in $F(c',c')$ if there is a morphism $f: c \to c'$ and an element x'' in $F(c',c)$ such that $F(f,c)(x'') = x$ and $F(c',f)(x'') = x'$.*

If $\mathbb{P}$ is a preorder viewed as a category and F is a bifunctor from $\mathbb{C}$ to $\mathbb{P}$, then the coend of F is computed as a join. We get the formula $\int^c F(c,c) = \bigvee_{c\in\mathbb{C}} F(c,c)$.

There are a few formulas to compute ends and coends, which are sometimes called "end and coend calculus". Here are two such formulas:

Proposition 2.2.18. *If F is a bifunctor from $\mathbb{C}$ to $\mathbb{D}$, then the following holds:*

- $\int_c [d, F(c,c)] \cong \left[d, \int_c F(c,c)\right]$,
- $\int_c [F(c,c), d] \cong \left[\int^c F(c,c), d\right]$.

There is also a formula to make ends commute:

Proposition 2.2.19 (Fubini theorem)**.** *If $F{:}\, \mathbb{C}^{op} \times \mathbb{D}^{op} \times \mathbb{C} \times \mathbb{D} \to \mathbb{E}$ is a functor, then if either side of $\int_c \int_d F(c,d,c,d) = \int_d \int_c F(c,d,c,d)$ exists, the equality holds.*

There are other interesting formulas, but we leave them for the next section, since they will appear more naturally in the framework of Kan extensions.

2.2.4 Kan Extensions

Required: 2.2.3.
Recommended: $\varnothing$.

The Kan extension of a functor $F{:}\,\mathbb{C} \to \mathbb{D}$ along $K{:}\,\mathbb{C} \to \mathbb{C}'$ can be thought of as a canonical way of extending F to a functor from $\mathbb{C}'$ to $\mathbb{D}$.

Definition 2.2.20. *Let $\mathbb{C}$, $\mathbb{C}'$ and $\mathbb{D}$ be categories, and $F{:}\,\mathbb{C} \to \mathbb{D}$ and $\mathbb{K}{:}\,\mathbb{C} \to \mathbb{C}'$ be functors. The* left *(resp.* right*)* Kan extension *of F along K, denoted by $\mathrm{Lan}_K(F)$ (resp. $\mathrm{Ran}_K(F)$), is a functor from $\mathbb{C}'$ to $\mathbb{D}$ equipped with a natural transformation $\eta{:}\, F \to \mathrm{Lan}_K(F)K$ (resp. $\varepsilon{:}\,\mathrm{Ran}_K(F)K \to F$) that is universal in the sense that, for all functors $G{:}\,\mathbb{C}' \to \mathbb{D}$ and natural transformations $\lambda{:}\, F \to GK$ (resp. $\lambda{:}\, GK \to F$), λ factors uniquely as $\lambda = (\mu{\cdot}K)\circ\eta$ (resp. $\lambda = \varepsilon \circ (\mu{\cdot}K)$).*

Under some hypotheses on $\mathbb{D}$ which we now describe, Kan extensions can be expressed in terms of ends and coends, so we can compute concrete values of Kan extensions.

Definition 2.2.21. *A category $\mathbb{C}$ is* powered *[60] (or* cotensored*) if, for all objects c of $\mathbb{C}$ and sets X, there is an object c^X of $\mathbb{C}$ equipped with a natural isomorphism $\mathbb{C}(-, c^X) \cong \mathsf{Set}(X, \mathbb{C}(-, c))$.*

Dually, a category $\mathbb{C}$ is copowered *if, for all objects c of $\mathbb{C}$ and set X, there is an object $X \cdot c$ of $\mathbb{C}$ equipped with a natural isomorphism $\mathbb{C}(X \cdot c, -) \cong \mathsf{Set}(X, \mathbb{C}(c, -))$.*

These definitions actually make sense in the broader sense of enriched category theory, but we only state them in the case of category theory, since we will only need the results in this case.

Example 2.2.22. *The category* Set *is powered and copowered. The power object Y^X is the set of functions from X to Y. The copower object $X \cdot Y$ is the cartesian product $X \times Y$.*

The category 2, which is defined as the ordinal $0 \to 1$ seen as a category, is also powered and copowered. The power objects are defined as:

$$\begin{cases} 0^{\varnothing} = 1 \\ 0^X = 0 & \textit{otherwise} \end{cases} \qquad\qquad 1^X = 1.$$

The copower objects are defined as:

$$X \cdot 0 = 0 \qquad\qquad \begin{cases} \varnothing \cdot 1 = 0 \\ X \cdot 1 = 1 & \textit{otherwise.} \end{cases}$$

The following property can be found, for example, in [76].

Property 2.2.23. *If $\mathbb{D}$ is powered, the right Kan extension of F along K is given by the formula $\mathrm{Ran}_K(F)(c') = \int_{c \in \mathbb{C}} F(c)^{\mathbb{C}'(c', Kc)}$.*

Similarly, if $\mathbb{D}$ is copowered, the left Kan extension of F along K is given by the formula $\mathrm{Lan}_K(F)(c') = \int^{c \in \mathbb{C}} \mathbb{C}'(Kc, c') \cdot F(c)$.

Notation 2.2.24. *When $F : \mathbb{C} \to \mathbb{D}$ is a functor and X is a presheaf over $\mathbb{C}$, we denote by $\sum_F(X)$ the left Kan extension of X along F^{op} and by $\prod_F(X)$ its right Kan extension along F^{op}.*

Using the descriptions of ends and coends in Set, we may give explicit descriptions of $\sum_F(X)$ and $\prod_F(X)$.

Proposition 2.2.25. *The left Kan extension $\sum_F(X)$ is given at d by the coend $\sum_F(X)(d) \cong \int^c X(c) \times [d, Fc]$, which is the coproduct $\sum_c X(c) \times [d, Fc]$ quotiented by the equivalence relation generated by equating $(x \cdot f, g)$ (with x in $X(c)$ and $g : d \to Fc$) and $(x, (Ff)g)$ for all $f : c \to c'$.*

We may recast this description in a more graphical way by depicting elements of $\sum_F(X)(d)$ as pairs of arrows $1 \to X(c)$ and $Fc \leftarrow d$, and the equivalence relation on elements is generated by "commuting diagrams" of the form:

$$\begin{tikzcd}
& 1 \arrow[dl, "\ulcorner x \cdot f \urcorner"'] \arrow[dr, "\ulcorner x \urcorner"] & \\
X(c) & & X(c') \arrow[ll, "X(f)"'] \\
F(c) \arrow[rr, "F(f)"'] & & F(c') \\
& d. \arrow[ul, "g"] \arrow[ur, "F(f)g"'] &
\end{tikzcd}$$

Proposition 2.2.26. *The expression of the right Kan extension $\prod_F(X)$ at d is $\prod_F(X)(d) \cong \int_c [[Fc, d], X(c)]$, which we recognise to be the set of natural transformations from the presheaf $[F-, d]$ to X.*

Lemma 2.2.27. *For any discrete fibration $p : \mathbb{E} \to \mathbb{B}$, presheaf X over $\mathbb{E}$, and object b in $\mathbb{B}$, we have $\sum_p(X)(b) \cong \sum_{p(e)=b} X(e)$, where $\sum$ means left Kan extension on the left and disjoint union on the right.*

Proof. We know that $\sum_p(X)$ is given at b by $\sum_p(X)(b) \cong \int^e X(e) \times [b, p(e)]$. But p is a discrete fibration, so all morphisms $g: b \to p(e)$ have a cartesian lifting $e \uparrow g: (e \cdot g) \to e$, so each pair of an element of $X(e)$ and morphism $g: b \to p(e)$ is equated in $\sum_p(X)(b)$ with a pair whose second member is an identity by:

$$\begin{tikzcd}
 & 1 \arrow[dl, "\ulcorner x \urcorner"'] \arrow[dr, "\ulcorner x \cdot_X (e\uparrow g) \urcorner"] & \\
X(e) \arrow[rr, "X(e\uparrow g)"] & & X(e \cdot_p g) \\
p(e) & & p(e \cdot_p g) \arrow[ll, "p(e\uparrow g)"] \\
 & b. \arrow[ul, "g"] \arrow[ur, equal] &
\end{tikzcd}$$

Now, let us consider $(1 \xrightarrow{x_1} X(e_1), p(e_1) \xleftarrow{f_1} b)$ and $(1 \xrightarrow{x_2} X(e_2), p(e_2) \xleftarrow{f_2} b)$ that are identified in a single step in $\sum_p(X)(b)$, i.e., there is a "commuting diagram" of the form:

$$\begin{tikzcd}
 & 1 \arrow[dl, "\ulcorner x_1 \urcorner"'] \arrow[dr, "\ulcorner x_2 \urcorner"] & \\
X(e_1) \arrow[rr, "X(g)"] & & X(e_2) \\
p(e_1) & & p(e_2) \arrow[ll, "p(g)"] \\
 & b \arrow[ul, "f_1"] \arrow[ur, "f_2"'] &
\end{tikzcd}
\quad \text{or} \quad
\begin{tikzcd}
 & 1 \arrow[dl, "\ulcorner x_1 \urcorner"'] \arrow[dr, "\ulcorner x_2 \urcorner"] & \\
X(e_1) & & X(e_2) \arrow[ll, "X(g)"'] \\
p(e_1) \arrow[rr, "p(g)"'] & & p(e_2) \\
 & b. \arrow[ul, "f_1"] \arrow[ur, "f_2"'] &
\end{tikzcd}$$

We want to show that (x_2, f_2) is equated with $(x_1 \cdot_X (e_1 \uparrow f1), id_b)$ in a single step. Because (x_1, f_1) is identified in a single step with $(x_1 \cdot_X (e_1 \uparrow f_1), id_b)$ through the diagram shown above, we get the desired result in the right-hand case by simply composing g and $e_1 \uparrow f_1$. In the left-hand case, since (x_2, f_2) is equated with $(x_2 \cdot_X (e_2 \uparrow f_2), id_b)$ in one step, we simply need to show that $x_2 \cdot_X (e_2 \uparrow f_2) = x_1 \cdot_X (e_1 \uparrow f_1)$, which comes from the facts that $e_1 \uparrow f_1 = g(e_2 \uparrow f_2)$ (by uniqueness of cartesian lifting) and that $x_1 \cdot g = x_2$ (by commutation of the diagram above).

Therefore, if two elements are equated, by induction on the length of the zigzag that equate them, they are both identified with the same pair $(1 \to X(e), p(e) = b)$ in a single step. We thus have that two pairs $(1 \xrightarrow{x} X(e), p(e) = b)$ and $(1 \xrightarrow{x'} X(e'), p(e') = b)$ can only be identified if $e = e'$ and $x = x'$. We thus have that $\sum_p(X)(b)$ is isomorphic to the coproduct $\sum_{p(e)=b} X(e)$. □

We also have the dual property:

Lemma 2.2.28. *For any discrete opfibration $p: \mathbb{E} \to \mathbb{B}$, presheaf X over $\mathbb{E}$, and object b in $\mathbb{B}$, we have $\prod_p(X)(b) \cong \prod_{p(e)=b} X(e)$, where $\prod$ means right Kan extension on the left and product on the right.*

Proof. We have that $\prod_p(X)(b)$ is isomorphic to the set of natural transformations $[[p-, b], X]$, so we want to show that this set is isomorphic to $\prod_{p(e)=b} X(e)$.

First, we send any natural transformation $\alpha: [p-, b] \to X$ to the product $(\alpha_e(id_b))_{p(e)=b}$. To show that this mapping is injective, we show that α is determined by the product. To prove this, let us take $f{:}\,p(e) \to b$ and show that $\alpha_e(f)$ must necessarily be a value that only depends on the product. Because p is a discrete opfibration, f has a cartesian lifting $f \uparrow e{:}\, e \to (f \cdot_p e)$. Therefore, by naturality of α, we have that $\alpha_e(f) = \alpha_{f \cdot_p e}(id_b) \cdot_X (f \uparrow e)$.

Conversely, let us take a product $(x_e)_{p(e)=b}$ and define α to be the transformation given by α_e mapping $f{:}\,p(e) \to b$ to $x_{f \cdot_p e} \cdot_X (f \uparrow e)$. This mapping is necessarily injective, because $\alpha_e(id_b) = x_e$ for all antecedents e of b. It thus only remains to show that α is natural, i.e., that for all $f{:}\,p(e') \to b$ and $g{:}\,e \to e'$, $x_{f \cdot_p e'} \cdot_X ((f \uparrow e')g) = x_{(fp(g)) \cdot_p e} \cdot_X (fp(g) \uparrow e)$, which follows from uniqueness of cartesian lifting. □

Let us also give two simple examples of left and right Kan extensions related to game semantics to give some intuition about them. The basic idea is that left Kan extension is a sort of existential quantifier, while right Kan extension is a sort of universal one.

Example 2.2.29. *Let σ be a behaviour on (A, B), i.e., a prefix-closed set of views (we leave the exact setting we are working with ambiguous, but the reader may think of classical HON games if they want a precise setting). We denote by $\mathbb{V}_{A,B}$ the poset of views on the pair of arenas (A, B) and by $\mathsf{i}_{A,B}$ the embedding of $\mathbb{V}_{A,B}$ into $\mathbb{P}_{A,B}$. With these notations behaviours can also be seen as functors from $\mathbb{V}_{A,B}$ to 2 (the ordinal $0 \to 1$ seen as a category). Indeed, if a view v is accepted by σ, then we map it to 1, and to 0 otherwise, and functoriality ensures prefix-closedness.*

We can take the right Kan extension of σ along $\mathsf{i}^{op}_{A,B}$, which we denote by $\prod_{\mathsf{i}_{A,B}}(\sigma) = \mathrm{Ran}_{\mathsf{i}^{op}_{A,B}}(\sigma)$, and it is given at p by $\int_{v \in \mathbb{V}_{A,B}} \sigma(v)^{\mathbb{P}_{A,B}(\mathsf{i}_{A,B} v, p)}$. Since 2 is an order, the formula reduces to $\prod_{\mathsf{i}_{A,B}}(\sigma)(p) = \bigwedge_{v \in \mathbb{V}_{A,B}} \sigma(v)^{\mathbb{P}_{A,B}(\mathsf{i}_{A,B} v, p)}$. Since $0^{\varnothing} = 1^X = 1$ (for any set X) and $0^X = 0$ (for any non-empty X), the formula above reduces to $\prod_{\mathsf{i}_{A,B}}(\sigma)(p) = \bigwedge_{\mathsf{i}_{A,B} v \to p} \sigma(v)$. In other words, $\prod_{\mathsf{i}_{A,B}}(\sigma)$ accepts to play p if and only if σ accepts to play all views v of p. This is the innocence *condition (defined in Section 2.1.1) imposed on innocent strategies. $\prod_{\mathsf{i}_{A,B}}$ thus turns a behaviour into the corresponding innocent strategy.*

Let σ be a "strategy" on $\mathbb{P}_{A,B,C}$, i.e., a functor $\mathbb{P}^{op}_{A,B,C} \to 2$ (typically the parallel composition of a strategy on (A, B) and a strategy on (B, C)). We can take the left Kan extension of σ along $\pi^{op}_{A,C}{:}\,\mathbb{P}^{op}_{A,B,C} \to \mathbb{P}^{op}_{A,C}$, and it is given by $\sum_{\pi_{A,C}}(\sigma)(p) = \int^{u \in \mathbb{P}_{A,B,C}} \mathbb{P}_{A,C}(p, \pi_{A,C} u) \cdot \sigma(u)$. Again, since 2 is an order, this reduces to $\sum_{\pi_{A,C}}(\sigma)(p) = \bigvee_{u \in \mathbb{P}_{A,B,C}} \mathbb{P}_{A,C}(p, \pi_{A,C}(u)) \cdot \sigma(u)$. Since $\varnothing \cdot 1 = X \cdot 0 = 0$ (for any set X) and $X \cdot 1 = 1$ (for any non-empty X), the formula above reduces to $\sum_{\pi_{A,C}}(\sigma)(p) = \bigvee_{\pi_{A,C} u = p} \sigma(u)$, assuming that all morphisms $p \to \pi_{A,C} u$ can be factored as $p = \pi_{A,C} u_0 \xrightarrow{\pi_{A,C} f} \pi_{A,C} u$. In other words, $\sum_{\pi_{A,C}}(\sigma)$ accepts to play p if and only if σ accepts to play at least one interaction sequence u whose projection on (A, C) is p. This is typically the hiding *operation, which hides what happens on the middle arena when two strategies are composed (as defined in again Section 2.1.1).*

In the cases we are interested in, namely presheaves, if $F{:}\,\mathbb{C} \to \mathbb{D}$, then $\sum_F{:}\,\widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$, $\prod_F{:}\,\widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$, and $\Delta_F{:}\,\widehat{\mathbb{D}} \to \widehat{\mathbb{C}}$.

Since we will be especially interested in presheaves, the following lemmas will be of some importance to us. The following lemma is named after the classical

Yoneda lemma (Property 2.2.37).

Lemma 2.2.30 (Yoneda lemma)**.** *Let $\mathbb{C}$ be a category and X a presheaf on $\mathbb{C}$ (i.e., a functor $X{:}\,\mathbb{C}^{op} \to \mathsf{Set}$), then $X(c) \cong \int_{c'}[[c',c],X(c')]$.*

Lemma 2.2.31 (co-Yoneda lemma)**.** *Let $\mathbb{C}$ be a category and X a presheaf on $\mathbb{C}$, then $X(c) \cong \int^{c'} X(c') \times [c,c']$.*

Proof. The first lemma may be seen directly, because $[c',c] = \mathsf{y}_c(c')$, so the formula reduces to $\int_{c'}[\mathsf{y}_c(c'),X(c')]$, which is isomorphic to the set of natural transformations from y_c to X by Example 2.2.14, and thus to $X(c)$ by the classical Yoneda lemma. However, it may be interesting to remark that $\int_{c'}[[c',c],X(c')]$ is also the expression of right Kan extension of X along the identity of $\mathbb{C}^{op}$. But right Kan extension of a functor along the identity is always isomorphic to the given functor, whence the result.

This also helps us understand why the second formula is called the co-Yoneda lemma. Indeed, the formula simply exhibits X as the *left* Kan extension of itself along the identity, which is dual to what happens in the Yoneda lemma. □

2.2.5 Presheaf Categories

Required: $\varnothing$.
Recommended: $\varnothing$.

Presheaves are so important in this dissertation that they definitely deserve their own section. We will use them extensively to model both plays and strategies. The intuition is that we want plays to be some kind of higher-dimensional graphs and strategies to be some kind of trees, and both graphs and trees are straightforward examples of presheaf categories.

Definition and Basic Properties

Definition 2.2.32 (Presheaf)**.** *A set-valued* [1] presheaf *over the category $\mathbb{C}$ is a functor from $\mathbb{C}^{op}$ to Set, the category of sets and functions.*

Presheaves over $\mathbb{C}$ and natural transformations form a category $[\mathbb{C}^{op},\mathsf{Set}]$ *that we denote by $\widehat{\mathbb{C}}$.*

Notation 2.2.33. *For any presheaf X in $\widehat{\mathbb{C}}$, $x \in X(c)$ and $f{:}\,c' \to c$, we denote by $x{\cdot}f$ the action of $X(f)$ on x, i.e., $x{\cdot}f = X(f)(x)$. Note that, by functoriality, we have that $(x \cdot f) \cdot g = x \cdot (fg)$.*

A very important property of the category of presheaves is the following:

Property 2.2.34. *If $\mathbb{C}$ is small, then $\widehat{\mathbb{C}}$ is complete and cocomplete, and limits and colimits are computed pointwise.*

In other terms, to compute the (co)limit of a functor $F{:}\,J \to \widehat{\mathbb{C}}$, we think of it as a functor $J \times \mathbb{C}^{op} \to \mathsf{Set}$ and compute the (co)limit for c fixed in $\mathbb{C}$, and this assignment uniquely extends to a presheaf that is the desired (co)limit. Or,

[1] We will always assume that presheaves are set-valued in the following, except when explicitly stated otherwise, so we call set-valued presheaves simply presheaves.

more concisely, $(\lim F)(c) \cong \lim(Fc)$. (This comes from the more general fact that, if $\mathbb{D}$ has (co)limits of a certain shape J, then so does $[\mathbb{C}, \mathbb{D}]$, and these (co)limits are computed pointwise, as explained in [76].)

This property is very important for us because it allows us to combine the objects we model as presheaves using limits and colimits. For example, in the case of plays, we can use constructions that "glue" further plays together to create different plays, and this is done using colimits (especially pushouts). Composition of plays is, in particular, defined using pushouts.

Definition 2.2.35 (Yoneda embedding)**.** *Let $\mathbb{A}$ be a category. We define the* Yoneda embedding *of $\mathbb{A}$ into $\widehat{\mathbb{A}}$ as the functor y whose value on a is $\mathsf{y}_a = [-, a]$, and which acts on morphisms by pre-composition.*

Note that the Yoneda embedding is indeed an embedding.

Property 2.2.36. *The Yoneda embedding y is full.*

Property 2.2.37 (Yoneda lemma)**.** *For all objects a of $\mathbb{A}$ and presheaf X over $\mathbb{A}$, there is an isomorphism $[\mathsf{y}_a, X] \cong X(a)$ natural in a.*

A nice, visual way of representing presheaves is through their categories of elements.

Definition 2.2.38 (Category of Elements)**.** *For any presheaf X on $\mathbb{A}$, we define X's* category of elements *as the comma category*

$$\begin{tikzcd}
\mathrm{el}(X) \arrow[r, "!"] \arrow[d, "\pi_X"'] \arrow[dr, phantom, "\overset{\lambda}{\Longrightarrow}"] & 1 \arrow[d, "\ulcorner X \urcorner"] \\
\mathbb{A} \arrow[r, "\mathsf{y}"'] & \widehat{\mathbb{A}}
\end{tikzcd}$$

together with its projection $\pi_X \colon \mathrm{el}(X) \to \mathbb{A}$.

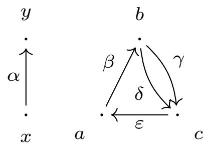
Recall from the definition of comma categories (Section 2.2.1) that an object of $\mathrm{el}(X)$ is a pair (a, x) of an object a of $\mathbb{A}$ and an element $x \in X(a)$, and that morphisms from (a, x) to (a', x') are exactly morphisms $f \colon a \to a'$ such that $x' \cdot f = x$. The projection π_X projects an element (a, x) onto its first component a. Even though it may be a bit mysterious why categories of elements are visual way of representing presheaves for now, the examples below should convince the reader that it is indeed the case.

Examples: Graphs and Trees

We now want to show how this notion adequately models the classical notions of graphs and trees.

Modelling graphs will give us the intuition of how to model plays in our games, since they are some sort of higher-dimensional graphs.

Example 2.2.39 (Graph: Running Example)**.** *Let us be more precise about what we mean by "graph", since there are many different notions of graphs. What we call graph is what may more accurately be called directed multi-graph, i.e., edges have a direction, and there may be more than one edge between two vertices. Let us take the graph below as our running example.*



Formally, a graph is a set of vertices, a set of edges, and two maps from the set of edges to the set of vertices that map each edge to its source and target respectively. For example, the drawing above is a representation of a graph with five vertices $V = \{x, y, a, b, c\}$*, five edges* $E = \{\alpha, \beta, \gamma, \delta, \varepsilon\}$*, and two maps* $s, t\colon E \to V$ *that map each edge to their source and target respectively, as on the drawing (for example,* $s(\alpha) = x$ *and* $t(\alpha) = y$*).*

The formal representation of graphs is very easily cast as a presheaf category:

Example 2.2.40 (Graphs)**.** *Let* $\mathbb{G}$ *be the category freely generated by the graph* $[0] \overset{s}{\underset{t}{\rightrightarrows}} [1]$*. The category of graphs is* $\widehat{\mathbb{G}}$*.*

Indeed, a presheaf G *over* $\mathbb{G}$ *is a functor* $\mathbb{G}^{op} \to \mathsf{Set}$*, i.e., two sets* $V = G([0])$ *and* $E = G([1])$ *together with two functions* $G(s), G(t)\colon V \to E$*. That is, a presheaf* G *is simply a set of vertices* V *and a set of edges* E *together with two maps that give, for each edge* e *in* E*, its source* $G(s)(e)$ *and target* $G(t)(e)$*.*

Moreover, a morphism of presheaves α *from* G *to* H *is a natural transformation from* G *to* H*, i.e., a pair of functions* $\alpha_0\colon G([0]) \to H([0])$ *and* $\alpha_1\colon G([1]) \to H([1])$ *such that both diagrams below commute.*

$$\begin{tikzcd}
G([1]) \arrow[r, "\alpha_1"] \arrow[d, "G(s)"'] & H([1]) \arrow[d, "H(s)"] \\
G([0]) \arrow[r, "\alpha_0"'] & H([0])
\end{tikzcd}
\qquad
\begin{tikzcd}
G([1]) \arrow[r, "\alpha_1"] \arrow[d, "G(t)"'] & H([1]) \arrow[d, "H(t)"] \\
G([0]) \arrow[r, "\alpha_0"'] & H([0])
\end{tikzcd}$$

In other words, such a natural transformation is a map of the vertices of G *to those of* H *and a map to the edges of* G *to those of* H *that respect sources and targets. This is exactly the definition of a morphism of graphs.*

Once we have modelled graphs as presheaves, we can instantiate the general theory of presheaves to graphs, for example the Yoneda lemma:

Example 2.2.41 (Graphs and the Yoneda Lemma)**.** *In the case of graphs, the Yoneda lemma simply states that the number of nodes in a graph* G *is equal to the number of morphisms from the graph* $\mathsf{y}_{[0]}$ *to* G*, and similarly for edges: the number of edges in* G *is equal to the number of morphisms from the graph* $\mathsf{y}_{[1]}$ *to* G*. The question is: What are* $\mathsf{y}_{[0]}$ *and* $\mathsf{y}_{[1]}$*? The former is the presheaf* $\mathbb{G}(-, [0])$*, i.e.,* $\mathsf{y}_{[0]}([0]) = \mathbb{G}([0], [0]) = \{id_{[0]}\}$ *and* $\mathsf{y}_{[0]}([1]) = \mathbb{G}([1], [0]) = \varnothing$*, so it is the graph with one node and no edge. The second one is* $\mathbb{G}(-, [1])$*, i.e.,* $\mathsf{y}_{[1]}([0]) = \mathbb{G}([0], [1]) = \{s, t\}$ *and* $\mathsf{y}_{[1]}([1]) = \mathbb{G}([1], [1]) = \{id_{[1]}\}$*, so it is a graph with two nodes and one edge, and the action of morphisms show which nodes are the source and target of the edge:* $\mathsf{y}_{[1]}(s)(id_{[1]}) = id_{[1]} \circ s = s$ *and* $\mathsf{y}_{[1]}(t)(id_{[1]}) = id_{[1]} \circ t = t$*. We thus have that* $\mathsf{y}_{[0]}$ *is the graph* $\cdot$ *and* $\mathsf{y}_{[1]}$ *is the graph* $\cdot \longrightarrow \cdot$*. In other words, the Yoneda lemma simply states that there are as many nodes in a graph* G *as there are morphisms from the "one-node graph" to* G *and as many edges in* G *as there are morphisms from the "one-edge graph" to* G*.*

Finally, the example of graphs is enlightening as to why the category of elements is a nice representation of a presheaf:

Example 2.2.42 (Graphs and Categories of Elements)**.** *Let G be a presheaf over $\mathbb{G}$. Its category of elements* $\mathrm{el}(G)$ *has as objects* $([0],x)$ *for all x in* $G([0])$ *and* $([1],y)$ *for all y in* $G([1])$. *Since, in the category of elements, morphisms* $(a,x) \to (a',x')$ *are simply the morphisms* $f\colon a \to a'$ *such that* $x' \cdot f = x$ *and the base category $\mathbb{G}$ only has two non-trivial morphisms, we have that, except for identities, the morphisms in* $\mathrm{el}(G)$ *are exactly of two possible forms: either* $([0],e \cdot s) \overset{s}{\to} ([1],e)$ *or* $([0],e \cdot t) \overset{t}{\to} ([1],e)$ *for any edge e.*

For example, the category of elements of the graph $\mathsf{y}_{[1]}$ *is*

$$\begin{tikzcd}
([0],s) \arrow[r, "s"] & ([1],id_{[1]}) & ([0],t) \arrow[l, "t"']
\end{tikzcd}$$

(where we omit the identities for readability). If we decide to draw elements of the form $([1],-)$ *and the corresponding pair of arrows* (s,t) *as an arrow pointing towards the source of t, we recover the graphic representation of the graph we started from:*

$$\begin{tikzcd}
([0],s) \arrow[r] & ([0],t),
\end{tikzcd}$$

which is indeed the graph with only one arrow.

Similarly, the category of elements of our running example is

$$\begin{tikzcd}
([0],y) \arrow[dd, "t"'] & & & & ([0],b) \arrow[ddl, "t"'] \arrow[dddr, "s"'] \arrow[drr, "s"] & & \\
& & & & & & ([1],\gamma) \\
([1],\alpha) & & & ([1],\beta) & & & \\
& & & & & ([1],\delta) & \\
([0],x) \arrow[uu, "s"] & & ([0],a) \arrow[uur, "s"] \arrow[rr, "t"'] & & ([1],\varepsilon) & & ([0],c), \arrow[uuu, "t"'] \arrow[ul, "t"] \arrow[ll, "s"]
\end{tikzcd}$$

from which we readily recover the description of the graph, using the same notation as above.

We may elaborate a bit further on this example by tweaking the base category. We can for example add an arrow $r\colon [1] \to [0]$ to $\mathbb{G}$ and equations $rt = id_{[0]}$ and $rs = id_{[0]}$ to obtain the category of (directed multi) reflexive graphs. The following extension is of particular interest to us because it is a way to define higher-dimensional graphs (even though our notion of higher-dimensional graph will be based on a slightly different approach):

Example 2.2.43 (Globular Sets)**.** *Let* $\mathbb{G}_\infty$ *be the category freely generated by the graph with a vertex $[n]$ for each natural number* $n \in \mathbb{N}$, *and edges* $s_n, t_n\colon [n] \to [n+1]$, *quotiented by the equations*

$$s_{n+1}s_n = t_{n+1}s_n \qquad \text{and} \qquad s_{n+1}t_n = t_{n+1}t_n.$$

We call $\mathbb{G}_\infty$ *the* globe category*. A* globular set *is a presheaf on* $\mathbb{G}_\infty$. *In simpler terms, a globular set G is a set of vertices* $G([0])$ *(which we call* 0-cells*), a set of edges* $G([1])$ *(which we call* 1-cells*), a set* $G([2])$ *of 2-cells, etc, and for* *each $(n+1)$-cell a source n-cell and a target n-cell, with the condition that the source and target n-cells agree on their own sources and targets. 2-cells can thus be thought of as "edges between parallel edges", 3-cells as "edges between parallel 2-cells", and so on, which is for example the type of n-morphisms in some approaches to higher category theory or the type of path homotopies.*

A second example of what may naturally be modelled by presheaves is that of trees, which will also be useful to model plays, since our plays are also trees in some sense. (We will always assume that trees are rooted, i.e., that they have a distinguished root.)

We show two possibilities to represent trees as presheaves. The first one is slightly more natural, but not so relevant to us, since this is not how we will represent our plays:

Example 2.2.44 (Trees, first presentation)**.** *Let ω be the countable ordinal category. A* forest *(of rooted trees) is a presheaf over ω. Indeed, such a presheaf F has for each $n \in \mathbb{N}$ a set $F(n)$ of nodes at depth n with, for each such node $x \in F(n+1)$, a parent node $x \cdot s_n$ (where s_n is the only morphism from n to $n+1$), which is indeed a forest. A tree is a presheaf T over ω such that $T(0)$ is a singleton set. The category $\widehat{\omega}$ is sometimes known as the* topos of trees*, which is somewhat fitting, since all presheaf categories are toposes (though a more fitting name would probably be the topos of forests).*

A natural transformation $\alpha \colon T \to T'$ is a family of maps $\alpha_n T(n) \to T'(n)$ for all $i \in \mathbb{N}$ such that, for all $n \in \mathbb{N}$,

$$\begin{tikzcd}
T(n+1) \arrow[r, "\alpha_{n+1}"] \arrow[d, "T(s_n)"'] & T'(n+1) \arrow[d, "T(s_{n+1})"] \\
T(n) \arrow[r, "\alpha_n"'] & T'(n)
\end{tikzcd}$$

commutes. In other words, it is a mapping of the nodes of T to those of T' that respects depth ($T(n)$ is mapped to $T'(n)$) and the parent relation, which is a possible, natural definition of morphism of forests. Similarly, a natural transformation between trees is a possible, natural definition of morphism of trees.

Here again, once we have defined trees (or forests) as presheaves on a particular category ω, the whole presheaf-theoretic lemmas give us some properties:

Example 2.2.45 (Trees and the Yoneda lemma)**.** *Here, the presheaf y_n is given by $\mathsf{y}_n(m) = \begin{cases} \{m \leq n\} & \textit{if } m \leq n \\ \varnothing & \textit{otherwise,} \end{cases}$ where $m \leq n$ is the unique morphism from m to n, when m is smaller than or equal to n, so $\mathsf{y}_n(m)$ is either a singleton or empty. Since each $\mathsf{y}_n(m)$ is either empty or a singleton set, there is only one possible action for morphisms. The presheaf y_n is thus a tree (since it has a single root $\mathsf{y}_n(0) = \{0 \leq n\}$) with a single node at depth m for all m less than or equal to n, and is empty below depth n. It is thus a branch of length n. The Yoneda lemma now states that there are as many morphisms of trees from the branch of length n into a forest F as there are nodes at depth n in F.*

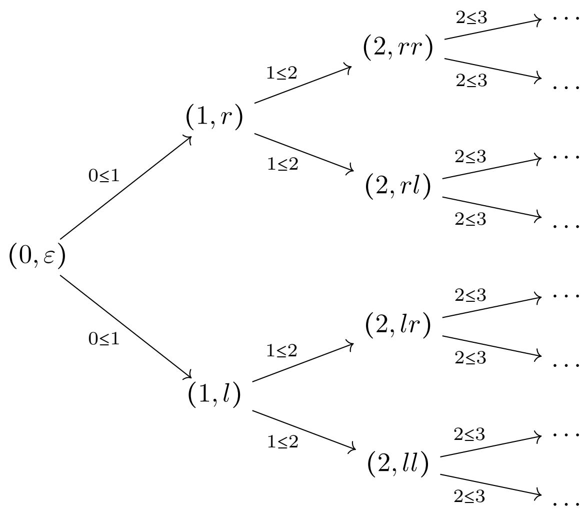
Finally, this presentation of trees also allows us to show once more why categories of elements are nice visual representations of presheaves:

Example 2.2.46 (Trees and Categories of Elements)**.** *Let us compute a few categories of elements. For example, let us take* y_n*. As we have seen above,* $\mathsf{y}_n(m)$ *is a singleton set* $\{m \leq n\}$ *if* m *is smaller than or equal to* n *and empty otherwise, with the only possible action for morphisms. Its category of elements thus has as elements all the pairs* $(m, m \leq n)$ *for* m *smaller than or equal to* n *and is as pictured below:*

$$\begin{tikzcd}
(0, 0 \leq n) \arrow[r, "0\leq 1"] & (1, 1 \leq n) \arrow[r, "1\leq 2"] & \dots \arrow[r, "n-1\leq n"] & (n, n \leq n)
\end{tikzcd}$$

(where we only show the $m \leq m+1$ *maps, and the other maps are obtained by composition). We recover the graphical representation of a branch.*

Let us now take the full infinite binary tree T*. This tree can straightforwardly be encoded as a presheaf over* ω *by taking* $T(n) = \{l, r\}^n$ *and where the action of a morphism* $m \leq n$ *is given by prefix:* $T(m \leq n)(w) = w_{|m}$*. Therefore, in the category of elements, there will be a morphism* $(m, w) \to (n, w')$ *exactly when* w *is a prefix of* w'*. If we draw this category of elements, we obtain:*



(where, once again, we only picture the $m \leq m+1$ *morphisms). Here again, we recover the graphic representation of the object through its category of elements.*

The second possibility is the one that will be more relevant to us. It consists in inductively building trees as particular graphs:

Example 2.2.47 (Trees, second presentation)**.** Trees *are particular pointed presheaves over* $\mathbb{G}$ *(elements of the coslice category* $\mathsf{y}_{[0]}/\widehat{\mathbb{G}}$*), i.e., particular morphisms* $\mathsf{y}_{[0]} \to T$ *in* $\widehat{\mathbb{G}}$*. In this description,* T *represents the "unrooted" tree structure and the morphism itself points to the root (remember that, from the Yoneda lemma, a morphism* $\mathsf{y}_{[0]} \to T$ *is the same as an element of* $T([0])$*). They are (isomorphic to any presheaf) defined inductively by:*

- *a single node* $id_{\mathsf{y}_{[0]}}\colon \mathsf{y}_{[0]} \to \mathsf{y}_{[0]}$ *is a tree,*
- *if* $r\colon \mathsf{y}_{[0]} \to T$ *is a tree and* $n\colon \mathsf{y}_{[0]} \to T$ *is a node of* T*, then* $fr\colon \mathsf{y}_{[0]} \to T'$ *is a tree, where* T' *and* f *are obtained by pushout as below.*

$$\begin{tikzcd}
 & \mathsf{y}_{[0]} \arrow[r, "s"] \arrow[d, "n"'] & \mathsf{y}_{[1]} \arrow[d] \\
\mathsf{y}_{[0]} \arrow[r, "r"'] & T \arrow[r, "f"'] & T' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

The inductive case in the definition above simply states:

- *that if T is a tree, by picking a node n of T (represented by the morphism $n{:}\mathsf{y}_{[0]} \to T$) and adding an edge to T whose source is n, the resulting graph is still a tree;*
- *and that all trees can be built in this way (so the category we are building is actually that of finite trees).*

A morphism of trees is a morphism in $\mathsf{y}_{[0]}/\widehat{\mathbb{G}}$, i.e., a morphism from $r{:}\mathsf{y}_{[0]} \to T$ to $r'{:}\mathsf{y}_{[0]} \to T'$ is a morphism $f{:}T \to T'$ such that

$$\begin{tikzcd}
 & \mathsf{y}_{[0]} \arrow[dl, "r"'] \arrow[dr, "r'"] & \\
T \arrow[rr, "f"'] & & T'
\end{tikzcd}$$

commutes. In other words, it a morphism of graphs that preserves the root, which is exactly a morphism of trees.

This example can also be tinkered with by adding different kinds of nodes and edges, which is something we do when we model our plays.

Remark. *Since plays are finite objects (they are basically execution traces), the finiteness of the objects we build is actually a feature, rather than a limitation. Had we wanted to model all trees (and not only finite ones) using the second method, we could have relied on transfinite composition by adding the following rule:*

- *for all limit ordinals β and cocontinuous functors $F{:}\beta \to \mathsf{y}_{[0]}/\widehat{\mathbb{G}}$ such that $F(\alpha) = \mathsf{y}_{[0]} \xrightarrow{r_\alpha} T_\alpha$ is a tree for all α, the morphism $r{:}\mathsf{y}_{[0]} \to T$ obtained as part of the cocone associated to the colimit below is also a tree:*

$$\begin{tikzcd}
 & & \mathsf{y}_{[0]} \arrow[dll, "r_0"'] \arrow[dl, "r_1"] \arrow[dr, "r_\alpha"] \arrow[dd, dashed, "r"] & & \\
T_0 \arrow[r] \arrow[dr, dashed] & T_1 \arrow[r] \arrow[d, dashed] & \dots \arrow[r] & T_\alpha \arrow[r] \arrow[dl, dashed] & \dots \\
 & & T. & &
\end{tikzcd}$$

A final example that should give all the necessary intuition to understand the representation of plays as string diagrams is that of binary trees. Indeed, until now, the categories of trees contain trees with arbitrary branching: a node can have a set of children. We want to build a category of binary trees, i.e., each node has either zero or two children. We will proceed in a way that is similar to the second presentation of trees that we have given above.

Since our inductive rule adds one edge to a tree, we first need to change it so that it adds two edges from the same node. We define the *basic binary tree* B as the coproduct:

$$\begin{tikzcd}
2 \cdot \mathsf{y}_{[0]} \arrow[r, "2\cdot t"] \arrow[d, "\nabla"'] & 2 \cdot \mathsf{y}_{[1]} \arrow[d, "l_B"] \\
\mathsf{y}_{[0]} \arrow[r, "r_B"'] & B. \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{2.1}$$

It consists of a root (pointed by r_B) with two children that are leaves. We can now change the rule to:

- if $r \colon \mathsf{y}_{[0]} \to T$ is a binary tree and $n \colon \mathsf{y}_{[0]} \to T$ is a node of T, then $fr \colon \mathsf{y}_{[0]} \to T'$ is a binary tree, where T' and f are obtained by pushout as below.

$$\begin{tikzcd}
 & \mathsf{y}_{[0]} \arrow[r, "r_B"] \arrow[d, "n"'] & B \arrow[d] \\
\mathsf{y}_{[0]} \arrow[r, "r"'] & T \arrow[r, "f"'] & T' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

However, this cannot work, since we can apply the inductive rule to a node twice, so the trees built this way are not binary at all (this construction simply constrains all nodes in a tree to have an even number of children). To make sure we do not use the inductive rule on the same node twice, we must keep track of which nodes are leaves and which nodes already have children.

A simple idea to do this is to put two different types of nodes in the base category $\mathbb{G}$: one for leaves and one for nodes. However, this is a bad idea for two reasons.

- First, we would need to duplicate the type of edges: there must be a type of edges that point to a node and a type of edges that point to a leaf. The base category $\mathbb{G}$ would then look like

  $$\begin{tikzcd}
  e_N & e_L \\
  N \arrow[u, shift left, "s"'] \arrow[u, shift right, "t"] \arrow[ur, "t"] & L, \arrow[u, "s"']
  \end{tikzcd}$$

  where N stands for “node” and L for “leaf”. Therefore, the creation of trees becomes more involved: once we have found a leaf in the tree, we must change its type to that of a node, change the type of the arrow that points to that leaf to the type of an arrow that points to a node, and finally, we can take the pushout along $\mathsf{y}_N \to B$ (where B is the obvious counterpart of the B defined above in this setting). This construction is not nice in the sense that it cannot be computed using limits and colimits.

- But the really problematic point is that the change made in the base category has an impact on the notion of morphism: nodes can only be mapped to nodes, and leaves only to leaves. This is obviously not the notion of morphism we are interested in, so this construction is not what we should do.

The idea of pointed presheaves $\mathsf{y}_{[0]} \to T$ is that the morphism distinguishes the root of T from the other nodes. We are going to do something similar here by giving another morphism $Y \xrightarrow{l} T$ from the set X of leaves of T to T.

Example 2.2.48 (Binary trees)**.** *Let $l\colon 2 \cdot \mathsf{y}_{[0]} \to B$ be the composite $2 \cdot \mathsf{y}_{[0]} \xrightarrow{2 \cdot s} 2 \cdot \mathsf{y}_{[1]} \xrightarrow{l_B} B$, where B and l_B are defined in* (2.1)*. A* move *is any cospan $Y \xrightarrow{l} U \xleftarrow{r} X$ that is isomorphic to that obtained as a pushout*

$$\begin{tikzcd}
 & 2 \cdot \mathsf{y}_{[0]} \arrow[rrr] \arrow[dr, "l_B"] & & & Y \arrow[dr, "l"] & \\
 & & B \arrow[rrr] & & & U \\
\varnothing \arrow[uur] \arrow[urr] \arrow[rrr] \arrow[dr] & & & Z \arrow[uur] \arrow[urr] \arrow[dr] & & \\
 & \mathsf{y}_{[0]} \arrow[uur, "r_B"] \arrow[rrr] & & & X, \arrow[uur, "r"'] &
\end{tikzcd}$$

where Z is a coproduct $n \cdot \mathsf{y}_{[0]}$, and l and r are obtained by universal property of pushout. We define the composition of two cospans $Y \to U \leftarrow X$ and $Z \to V \leftarrow Y$ as the cospan $Z \to U \bullet V \leftarrow X$ defined as

$$\begin{tikzcd}
 & & Z \arrow[d] \\
 & Y \arrow[r] \arrow[d] & V \arrow[d] \\
X \arrow[r] & U \arrow[r] & U \bullet V. \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

A forest of binary trees *is a cospan of presheaves $Y \to F \leftarrow X$ that is isomorphic to a finite composite of moves. A* binary tree *is a forest of binary trees of the form $Y \to F \leftarrow \mathsf{y}_{[0]}$.*

We now have at least two natural notions of morphism of forests.

- *If $Y \to F \leftarrow X$ and $Y' \to F' \leftarrow X'$ are two forests, then a morphism of forests is a natural transformation $F \to F'$. This yields the notion of morphism of (unrooted) forest, i.e., of morphisms that do not necessarily preserve the roots of forests.*
- *If $Y \to F \leftarrow X$ and $Y' \to F' \leftarrow X'$ are two forests, then a morphism of forests is a triple of natural transformations $X \to X'$, $F \to F'$, and $Y \to Y'$ such that*

$$\begin{tikzcd}
Y \arrow[r] \arrow[d] & Y' \arrow[d] \\
F \arrow[r] & F' \\
X \arrow[r] \arrow[u] & X' \arrow[u]
\end{tikzcd}$$

commutes. This yields a slightly strange notion of morphism of forests. Such a morphism preserves not only the roots of forests, but also their leaves.

However, the usual notion of morphism of rooted forests is given by a pair of morphisms $F \to F'$ and $X \to X'$ such that

$$\begin{tikzcd}
Y \arrow[d] & Y' \arrow[d] \\
F \arrow[r] & F' \\
X \arrow[u] \arrow[r] & X' \arrow[u]
\end{tikzcd}$$

commutes.

Equivalence to Discrete Fibrations

Required: 2.2.2, beginning of 2.2.5.
Recommended: $\varnothing$.

The adjoint equivalence between the category of presheaves over $\mathbb{A}$ and that of discrete fibrations into $\mathbb{A}$ is well known. However, we have found no clear and complete account of this fact, especially one that exhibits the equivalence, so we decided to write it here in case it would be of general interest.

We start by recalling a similar, simpler result:

Proposition 2.2.49. *For any set I, let Fam_I be the category of families of sets indexed by I, whose objects are families $\{X_i\}_{i\in I}$ and whose morphisms from $\{X_i\}_{i\in I}$ to $\{X'_i\}_{i\in I}$ are families of functions $\{f_i : X_i \to X'_i\}_{i\in I}$.* Fam_I *is equivalent to $[-, I]$.*

Example 2.2.50 (Hogwarts School of Witchcraft and Wizardry [2] [3])**.** *Hogwarts School of Witchcraft and Wizardry is divided into four houses:* Gryffindor, Hufflepuff, Ravenclaw, *and* Slytherin. *Each student at Hogwarts belongs to exactly one of these four schools. This may be represented in two equivalent ways:*

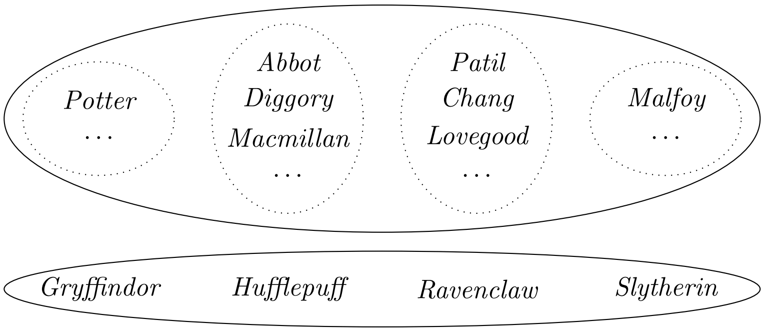
- *by giving a family of sets of students, indexed by houses:*
  $X_{\text{Gryffindor}} = \{\text{Harry Potter}, \text{Ronald Weasley}, \text{Hermione Granger}, \ldots\}$,
  $X_{\text{Hufflepuff}} = \{\text{Cedric Diggory}, \text{Ernie Macmillan}, \text{Hannah Abbot}, \ldots\}$,
  $X_{\text{Ravenclaw}} = \{\text{Luna Lovegood}, \text{Cho Chang}, \text{Padma Patil}, \ldots\}$,
  $X_{\text{Slytherin}} = \{\text{Draco Malfoy}, \text{Vincent Crabbe}, \text{Gregory Goyle}, \ldots\}$,

- *by giving a map from the set of students to that of houses:*
  Harry Potter ↦ Gryffindor*,*
  Ronald Weasley ↦ Gryffindor*,*
  Cedric Diggory ↦ Hufflepuff*,*
  ...

This can be represented by the drawing below, where the set of students is represented at the top and that of houses at the bottom.

[2] Thanks to Ed Morehouse, who gave this ludic example during OPLSS 2015.

[3] All Harry Potter© characters, names, and places belong to J. K. Rowling and are used under fair copyright law.



The map that assigns each student to their house is the "vertical projection" map, while the indexed family associates to each house the set "above" it.

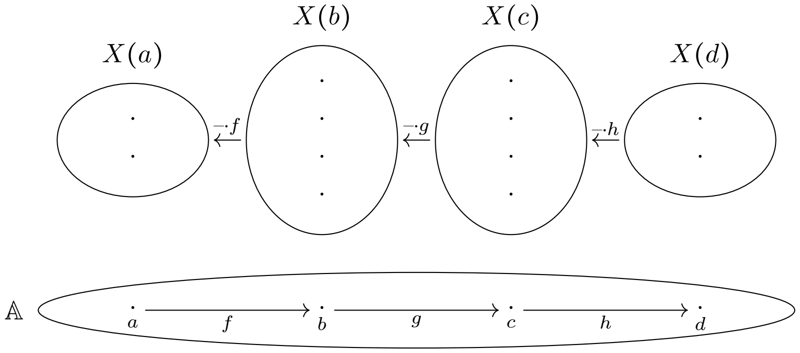
In the general case, the equivalence works as follows:

- to turn $f: X \to I$ into an indexed family, we organise X into the fibres of f (formally, f is mapped to the indexed family $\{f^{-1}(i)\}_{i\in I}$),
- to turn $\{X_i\}_{i\in I}$ into a map, we amalgamate all the different X_i's into one (formally, this family is mapped to the function from $\sum_{i\in I} X_i$ to I that maps (i, x) to i).

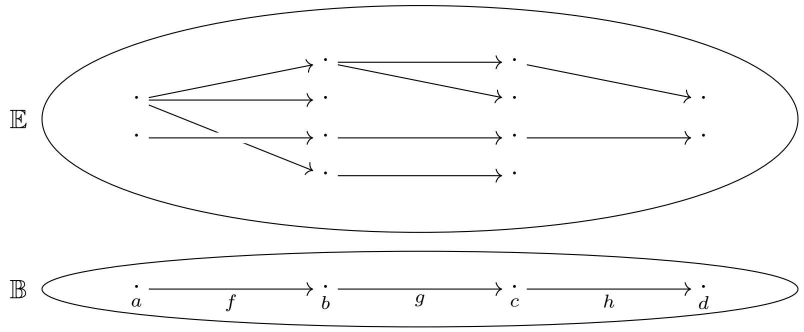
It is then simple to show that this forms an equivalence.

We now set out to prove the equivalence between the category of presheaves over $\mathbb{A}$ and that of discrete fibrations into $\mathbb{A}$. The intuition is the same as above, except that there are now morphisms involved: presheaves correspond to families indexed by I (a presheaf X maps each a in $\mathbb{A}$ to a set $X(a)$), while discrete fibrations into $\mathbb{A}$ correspond to maps into I.

A presheaf X associates to each a in $\mathbb{A}$ a set $X(a)$, and to each morphism $f: a \to a'$ a function $- \cdot f: X(a') \to X(a)$. This can be pictured as follows:



For $p: \mathbb{E} \to \mathbb{A}$ to be a discrete fibration means that for each $f: a \to a'$, we can define a map from the fibre $p^{-1}(a')$ to $p^{-1}(a)$ that maps x to $x \cdot_p f$, the domain of the lifting of x along f. If we want to represent this fact in a picture similar to the previous ones, it looks like the picture below. What should be noticed on this picture is that, for a given x in the fibre of p over, say, c, there is exactly one morphism over g whose codomain is x.



The idea is then simply to organise $\mathbb{E}$ into the fibres of p to recover a presheaf over $\mathbb{A}$, and conversely to turn a presheaf into a discrete fibration by amalgamating all the $X(a)$'s and the functions between them into a category: its category of elements.

Proposition 2.2.51. *For any presheaf X over $\mathbb{A}$, the projection $\pi_X\colon \mathrm{el}(X) \to \mathbb{A}$ is a discrete fibration.*

Proof. Take an object (a,x) of $\mathbb{A}$ and a morphism $f\colon a' \to \pi_X((a,x)) = a$. First, $\pi_X(f\colon (a', x \cdot_X f) \to (a,x)) = f\colon a' \to a$, so f has an antecedent through π_X. Furthermore, if $\pi_X(g\colon (a'', x'') \to (a,x)) = f\colon a' \to a$, then $g = f$ by definition of π_X, so π_X is indeed a discrete fibration. □

Conversely, to organise a discrete fibration p into its fibres, we use the following functor:

Definition 2.2.52. *Let $\partial\colon \mathbb{A} \to \mathsf{DFib}_{\mathbb{A}}$ be the functor that maps a to the domain functor $\partial_a\colon \mathbb{A}/a \to \mathbb{A}$ and $f\colon a \to a'$ to $f \circ -\colon \mathbb{A}/a \to \mathbb{A}/a'$.*

Proposition 2.2.53. *∂ is well-defined.*

Proof. We simply need to show that ∂_a is a discrete fibration, as the facts that ∂_f is a morphism of discrete fibrations and that ∂ is functorial are obvious. We take an object $f\colon a' \to a$ of $\mathbb{A}/a$, and a morphism $g\colon a'' \to a'$ of $\mathbb{A}$. We want to show that there is a unique morphism to f in $\mathbb{A}/a$ that is mapped to g by ∂_a. Since $g\colon fg \to f$ is such a morphism, existence ensues. Moreover, the only morphism that is mapped to g is g itself, which proves uniqueness. □

Definition 2.2.54 (Nerve Functor)**.** *For any functor $F\colon \mathbb{A} \to \mathbb{B}$, we define the* nerve functor *$F^\star\colon \mathbb{B} \to \widehat{\mathbb{A}}$ as:*

$$\begin{aligned}
F^\star(b)(a) &= [Fa, b] \\
F^\star(b)(f\colon a \to a') &= \left\{ \begin{array}{ccc} [Fa', b] & \to & [Fa, b] \\ u & \mapsto & u(Ff) \end{array} \right. \\
(F^\star(u\colon b \to b'))_a &= \left\{ \begin{array}{ccc} [Fa, b] & \to & [Fa, b'] \\ v & \mapsto & uv. \end{array} \right.
\end{aligned}$$

We think of ∂ as organising $\mathbb{X}$ into the fibres of $p\colon \mathbb{X} \to \mathbb{A}$ because:

Proposition 2.2.55. *For all discrete fibrations $p\colon \mathbb{X} \to \mathbb{A}$ and objects a of $\mathbb{A}$, $p^{-1}(a) \cong \partial^\star(p)(a)$, naturally in p and a.*

Proof. For all objects a of $\mathbb{A}$, $\partial^\star(p)(a)$ is the set of morphisms $f{:}\,\mathbb{A}/a \to \mathbb{X}$ such that

$$\begin{tikzcd}
\mathbb{A}/a \arrow[rr, "f"] \arrow[dr, "\partial_a"'] & & \mathbb{X} \arrow[dl, "p"] \\
& \mathbb{A} &
\end{tikzcd}$$

commutes, and $p^{-1}(a) = \{x \mid p(x) = a\}$.

Let us first show that every f in $\partial^\star(p)(a)$ is determined by its image on id_a. Let us define $x = f(id_a)$. We know by commutation of the triangle above that, for all $g{:}\,a' \to a$, $p(f(g)) = \partial_a(g) = a'$, and if we see g as a morphism $g{:}\,g \to id_a$ in $\mathbb{A}/a$, we have that $p(f(g)) = \partial_a(g) = g$. We therefore have that $f(g)$ is a cartesian lifting of x along g, hence, by uniqueness of the cartesian lifting, f is determined by $f(id_a)$.

Moreover, for any x such that $p(x) = a$, the mapping described above is functorial and makes the triangle commute, and therefore yields an f. Commutation of the triangle is obvious by definition. Furthermore, functoriality is given by "functoriality" of cartesian lifting (Proposition 2.2.7).

Both sets are thus isomorphic. Naturality in a and p is direct. □

We prefer using $\partial^\star$ because it makes diagrammatic reasoning easier.

We finally have:

Lemma 2.2.56. *The pair of functors* $\begin{tikzcd} \widehat{\mathbb{A}} \arrow[r, bend left, "\pi"] \arrow[r, phantom, "\perp"] & \mathsf{DFib}_{\mathbb{A}} \arrow[l, bend left, "\partial^\star"] \end{tikzcd}$ *is an adjoint equivalence.*

Proof. We need to define $\eta{:}\,\mathrm{Id}_{\widehat{\mathbb{A}}} \to \partial^\star\pi$ and $\varepsilon{:}\,\pi\partial^\star \to \mathrm{Id}_{\mathsf{DFib}_{\mathbb{A}}}$ and show that they are isomorphisms and satisfy the triangle identities.

By Lemma 2.2.55, morphisms of discrete fibrations from ∂_a to some p are in one-to-one correspondence with antecedents of a through p. Let us call $\overline{x}$ the morphism of discrete fibrations that corresponds to x through this isomorphism. Remember from the proof that $\overline{x}(id_a) = x$.

We define η by defining all its components η_X, which are again natural transformations, whose components are $\eta_{X,a}{:}\,X(a) \to [\partial_a, \pi_X]$, $\eta_{X,a}(x) = \overline{(a,x)}$. To prove naturality of η_X, we need to show that, for all $f{:}\,a' \to a$,

$$\begin{tikzcd}
Xa \arrow[r, "\eta_{X,a}"] \arrow[d, "-\cdot f"'] & {[\partial_a, \pi_X]} \arrow[d, "{[\partial_f, \pi_X]}"] \\
Xa' \arrow[r, "\eta_{X,a'}"'] & {[\partial_{a'}, \pi_X]}
\end{tikzcd}$$

commutes, which, because morphisms $\partial_{a'} \to \pi_X$ are determined by their image on $id_{a'}$, amounts to showing that $(a,x)\cdot_{\pi_X} f = (a', x \cdot_X f)$. But

$$\begin{tikzcd}
(a', x \cdot_X f) \arrow[rr, "f"] \arrow[d, mapsto, "\pi_X"'] & {} \arrow[d, mapsto, "\pi_X"] & (a,x) \arrow[d, mapsto, "\pi_X"] \\
a' \arrow[rr, "f"'] & {} & a
\end{tikzcd}$$

exhibits $(a', x \cdot_X f)$ as having the property of $(a,x) \cdot_{\pi_X} f$, so they are equal. Finally, to prove naturality of η, we need to show that for any $\alpha{:}\, X \to Y$,

$$\begin{tikzcd}
X \arrow[r, "\eta_X"] \arrow[d, "\alpha"'] & {[\partial, \pi_X]} \arrow[d, "{[\partial, \pi_\alpha]}"] \\
Y \arrow[r, "\eta_Y"'] & {[\partial, \pi_Y]}
\end{tikzcd}$$

commutes, which amounts to showing that

$$\begin{tikzcd}
X(a) \arrow[r, "\eta_{X,a}"] \arrow[d, "\alpha_a"'] & {[\partial_a, \pi_X]} \arrow[d, "{[\partial_a, \pi_\alpha]}"] \\
Y(a) \arrow[r, "\eta_{Y,a}"'] & {[\partial_a, \pi_Y]}
\end{tikzcd}$$

commutes for all a. It suffices to show that, for all x in $X(a)$, the functors we obtain are equal when applied to id_a, which is the case, since they both map it to $(a, \alpha_a(x))$.

We now define ε by defining its components ε_p as the functors:

$$\begin{array}{ccc}
\pi_{\partial^\star(p)} & \to & p \\
(a, g{:}\, \partial_a \to p) & \mapsto & g(id_a) \\
(f{:}\, (a', g(\mathbb{A}/f)) \to (a, g)) & \mapsto & g(id_a) \uparrow_p f.
\end{array}$$

Functoriality of ε_p is given by Proposition 2.2.7 (functoriality of cartesian lifting). Naturality of ε is given by compatibility of cartesian lifting with morphisms of discrete fibrations (Proposition 2.2.8).

We now show that η and ε are isomorphisms. We define $\eta^{-1}_{X,a}{:}\, [\partial_a, \pi_X] \to X(a)$ as the function that maps $f{:}\, \partial_a \to \pi_X$ to $f(id_a)$ and show that it is an inverse to $\eta_{X,a}$. In one direction, $(\eta^{-1}_{X,a} \circ \eta_{X,a})(x) = \eta^{-1}_{X,a}(\overline{(a,x)}) = \overline{(a,x)}(id_a) = x$. In the other, by recalling that two morphisms of discrete fibrations from ∂_a are equal if they agree on id_a, $(\eta_{X,a} \circ \eta^{-1}_{X,a})(f)(id_a) = \eta_{X,a}(f(id_a))(id_a) = \overline{(a, f(id_a))}(id_a) = f(id_a)$. For ε, we define ε_p^{-1} as the functor:

$$\begin{array}{ccc}
x & \mapsto & (p(x), \overline{x}) \\
(f{:}\, x \to y) & \mapsto & (p(f){:}\, (p(x), \overline{x}) \to (p(y), \overline{y}).
\end{array}$$

It is not difficult to show that ε_p^{-1} is well-defined, i.e., that $p(f)$ indeed has the desired domain and codomain. Now, we need to show that ε_p^{-1} is the inverse of ε_p. We first show it on objects. In one direction, by noticing that all objects in $\mathrm{el}(\partial^\star(p))$ are of the form $(p(x), \overline{x})$ for some object x in $\mathbb{X}$, we get $(\varepsilon_p^{-1} \circ \varepsilon_p)((p(x), \overline{x})) = \varepsilon_p^{-1}(\overline{x}(id_{p(x)})) = \varepsilon_p^{-1}(x) = (p(x), \overline{x})$. In the other direction, $(\varepsilon_p \circ \varepsilon_p^{-1})(x) = \varepsilon_p((p(x), \overline{x})) = \overline{x}(id_{p(x)}) = x$. We then show it on morphisms. In one direction, if f is a morphism from $(p(x), \overline{x})$ to $(p(y), \overline{y})$, then $(\varepsilon_p^{-1} \circ \varepsilon_p)(f) = \varepsilon_p^{-1}(\overline{y}(id_{p(y)}) \uparrow_p f) = \varepsilon_p^{-1}(y \uparrow_p f) = p(y \uparrow_p f) = f$ because the projection of a cartesian lifting along f is necessarily f by definition. In the other direction, if f is a morphism from x to y, then $(\varepsilon_p \circ \varepsilon_p^{-1})(f) = \varepsilon_p(p(f)) = \overline{y}(id_{p(y)}) \uparrow_p p(f) = y \uparrow_p p(f) = f$ by uniqueness of cartesian lifting.

Finally, we show that η and ε satisfy the triangle identities. For the first equality, we must show $(\partial^\star \cdot \varepsilon)(\eta \cdot \partial^\star) = \mathrm{Id}_{\partial^\star}$, for which it suffices to show that both transformations are equal on all components p, for which it again suffices to show that they are equal on all components p, a. We have $((\partial^\star \cdot \varepsilon)(\eta \cdot \partial^\star))_{p,a} = (\partial^\star(\varepsilon_p))_a \eta_{\partial^\star(p),a}$, and when applied to $f{:}\, \partial_a \to p$, $((\partial^\star \cdot \varepsilon)(\eta \cdot \partial^\star))_{p,a}(f) = (\partial^\star(\varepsilon_p))_a(\eta_{\partial^\star(p),a}(f)) = (\partial^\star(\varepsilon_p))_a(\overline{(a, f)}) = \varepsilon_p \circ \overline{(a, f)}$. Since morphisms of discrete fibrations from ∂_a are determined by their values on id_a, to show that this is equal to f, we just show that they are equal when applied to id_a. $((\partial^\star \cdot \varepsilon)(\eta \cdot \partial^\star))_{p,a}(f)(id_a) = \varepsilon_p(\overline{(a, f)}(id_a)) = \varepsilon_p((a, f)) = f(id_a)$. For the second identity, we must show that $(\varepsilon \cdot \pi)(\pi \cdot \eta) = \mathrm{Id}_\pi$, which is equivalent to showing that they are equal on all components. We have that $((\varepsilon \cdot \pi)(\pi \cdot \eta))_X = \varepsilon_{\pi_X} \pi_{\eta_X}$. To show that both transformations are equal on objects, we apply them to an object (a, x) and get $((\varepsilon \cdot \pi)(\pi \cdot \eta))_X((a, x)) = \varepsilon_{\pi_X}(\pi_{\eta_X}((a, x))) = \varepsilon_{\pi_X}((a, \overline{(a, x)})) = \overline{(a, x)}(id_a) = (a, x)$. On morphisms, if $f{:}\,(a', x') \to (a, x)$, then $((\varepsilon \cdot \pi)(\pi \cdot \eta))_X(f) = \varepsilon_{\pi_X}(\pi_{\eta_X}(f)) = \varepsilon_{\pi_X}(f{:}\,(a', \overline{(a', x')}) \to (a, \overline{(a, x)})) = \overline{(a, x)}(id_a) \uparrow_{\pi_X} f = (a, x) \uparrow_{\pi_X} f = f$. □

Restriction and Extensions

Required: 2.2.4, beginning of 2.2.5.
Recommended: ∅.

Remember that, if F is a functor from $\mathbb{C}$ to $\mathbb{D}$, then it induces three functors between $\widehat{\mathbb{C}}$ and $\widehat{\mathbb{D}}$. The first one is restriction along F^{op}, denoted by Δ_F. When $\mathbb{C}$ and $\mathbb{D}$ are small, this restriction functor admits left and right adjoints, respectively given by left and right Kan extension along F^{op}:

Lemma 2.2.57. *The following adjunctions hold*

$$\begin{tikzcd}
\widehat{\mathbb{C}} \arrow[rr, bend left=40, "\Sigma_F"] \arrow[rr, bend right=40, "\Pi_F"'] & \perp \\ \perp & \widehat{\mathbb{D}}. \arrow[ll, "\Delta_F" description]
\end{tikzcd}$$

Proof. For the top adjunction, it suffices to show that, for all presheaves X over $\mathbb{C}$ and Y over $\mathbb{D}$, $[\sum_F(X), Y] \cong [X, \Delta_F(Y)]$ naturally, which is given by the following computation:

$$\begin{aligned}
[\textstyle\sum_F(X), Y] &\cong \int_d [\textstyle\sum_F(X)(d), Y(d)] && \text{by Example 2.2.14} \\
&\cong \int_d \Big[\int^c X(c) \times [d, Fc], Y(d)\Big] \\
&\cong \int_d \int_c [X(c) \times [d, Fc], Y(d)] \\
&\cong \int_c \int_d [X(c), [[d, Fc], Y(d)]] && \text{by Fubini and copower} \\
&\cong \int_c \Big[X(c), \int_d [[d, Fc], Y(d)]\Big] \\
&\cong \int_c [X(c), Y(F(c))] && \text{by the Yoneda lemma} \\
&\cong \int_c [X(c), \Delta_F(Y)(c)] \\
&\cong [X, \Delta_F(Y)].
\end{aligned}$$

Since each step arises as a natural transformations, the whole computation is also natural.

The idea is the same for the bottom adjunction. Here is the computation:

$$\begin{aligned}
[Y, \textstyle\prod_F(X)] &\cong \int_d [Y(d), \textstyle\prod_F(X)(d)] \\
&\cong \int_d \left[Y(d), \int_c [[Fc, d], X(c)] \right] \\
&\cong \int_d \int_c [Y(d), [[Fc, d], X(c)]] \\
&\cong \int_c \int_d [Y(d) \times [Fc, d], X(c)] \\
&\cong \int_c \left[\int^d Y(d) \times [Fc, d], X(c) \right] \\
&\cong \int_c [Y(Fc), X(c)] && \text{by the co-Yoneda lemma} \\
&\cong \int_c [\Delta_F(Y)(c), X(c)] \\
&\cong [\Delta_F(Y), X].
\end{aligned}$$

□

Finally, it will be useful at some point to know what Δ_F and $\sum_F$ amount to when presheaves are translated to discrete fibrations (at least in some cases), which is what we explain next.

Lemma 2.2.58. *If $F\colon \mathbb{C} \to \mathbb{D}$ is a functor and Y is a presheaf over $\mathbb{D}$, then the functor $\tilde{F}\colon \mathrm{el}(\Delta_F(Y)) \to \mathrm{el}(Y)$ that maps (c, x) to (Fc, x) and $f\colon (c, x) \to (c', x')$ to $Ff\colon (Fc, x) \to (Fc', x')$ makes*

$$\begin{tikzcd}
\mathrm{el}(\Delta_F(Y)) \arrow[r, "\tilde{F}"] \arrow[d, "\pi_{\Delta_F(Y)}"'] & \mathrm{el}(Y) \arrow[d, "\pi_Y"] \\
\mathbb{C} \arrow[r, "F"'] & \mathbb{D}
\end{tikzcd}$$

a pullback.

Proof. The proof is direct:

- for objects, it suffices to show that an object c of $\mathbb{C}$ and an element (Fc, x) of $\mathrm{el}(Y)$ have a unique common antecedent in $\mathrm{el}(\Delta_F(Y))$, which is necessarily (c, x),
- for arrows, it suffices to show that a morphism $f\colon c \to c'$ of $\mathbb{C}$ and a morphism $F(f)\colon (Fc, x) \to (Fc', x')$ of $\mathrm{el}(Y)$ have a unique antecedent in $\mathrm{el}(\Delta_F(Y))$, which is necessarily $f\colon (c, x) \to (c', x')$. □

Lemma 2.2.59. *If $F\colon \mathbb{C} \to \mathbb{D}$ is a discrete fibration and X is a presheaf over $\mathbb{C}$, then $\pi_{\sum_F(X)}\colon \mathrm{el}(\sum_F(X)) \to \mathbb{D}$ is isomorphic to $F\pi_X$.*

Proof. We want to show that there is an isomorphism $\mathfrak{i}\colon \mathrm{el}(X) \to \mathrm{el}(\sum_F(X))$ such that

$$\begin{tikzcd}
\mathrm{el}(X) \arrow[r, "\mathsf{i}"] \arrow[d, "\pi_X"'] & \mathrm{el}(\sum_F(X)) \arrow[d, "\pi_{\sum_F(X)}"] \\
\mathbb{C} \arrow[r, "F"'] & \mathbb{D}
\end{tikzcd}$$

commutes. We know that elements of $\sum_F(X)$ are pairs of an object d of $\mathbb{D}$ and a pair $(1 \to X(c), F(c) \leftarrow d)$ quotiented by the usual triangle equation. We can thus define i as the functor that maps (c, x) to $(Fc, (\ulcorner x \urcorner, id_{Fc}))$ (and maps f to "pre-composition" by Ff). This functor is not invertible in the general case, but because F is a discrete fibration, by Lemma 2.2.27, we have that $\sum_F(X)(d) \cong \sum_{Fc=d} X(c)$, so it is indeed invertible. □

These results may be rephrased in the following terms: if we denote by $\Delta_F \colon \mathsf{DFib}_{\mathbb{D}} \to \mathsf{DFib}_{\mathbb{C}}$ the functor that pulls back along F and by $\sum_F \colon \mathsf{DFib}_{\mathbb{C}} \to \mathsf{DFib}_{\mathbb{D}}$ the functor that post-composes with F, then

$$\begin{tikzcd}
\widehat{\mathbb{C}} \arrow[d, "\pi"'] & \widehat{\mathbb{D}} \arrow[l, "\Delta_F"'] \arrow[d, "\pi"] \\
\mathsf{DFib}_{\mathbb{C}} & \mathsf{DFib}_{\mathbb{D}} \arrow[l, "\Delta_F"]
\end{tikzcd} \quad \text{and} \quad \begin{tikzcd}
\widehat{\mathbb{C}} \arrow[r, "\Sigma_F"] \arrow[d, "\pi"'] & \widehat{\mathbb{D}} \arrow[d, "\pi"] \\
\mathsf{DFib}_{\mathbb{C}} \arrow[r, "\Sigma_F"'] & \mathsf{DFib}_{\mathbb{D}}
\end{tikzcd}$$

commute up to isomorphism (the second one only on the condition that F be a discrete fibration).

2.2.6 Exact Squares

Required: 2.2.4,2.2.5.
Recommended: 2.2.1.

We will use Guitart's theory of *exact squares* [45] several times in this dissertation, so we explain it now.

Definition 2.2.60. *A* square *is a natural transformation*

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "T"] \arrow[d, "S"'] \arrow[dr, phantom, "\overset{\varphi}{\Longrightarrow}"] & \mathbb{B} \arrow[d, "V"] \\
\mathbb{C} \arrow[r, "U"'] & \mathbb{D}.
\end{tikzcd} \tag{2.2}$$

where $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$ and $\mathbb{D}$ are small categories and S, T, U, and V are functors.

Any square yields by restriction a square

$$\begin{tikzcd}
\widehat{\mathbb{A}} & \widehat{\mathbb{B}} \arrow[l, "\Delta_T"'] \arrow[dl, phantom, "\overset{\Delta_\varphi}{\Longleftarrow}"] \\
\widehat{\mathbb{C}} \arrow[u, "\Delta_S"] & \widehat{\mathbb{D}}. \arrow[l, "\Delta_U"] \arrow[u, "\Delta_V"']
\end{tikzcd}$$

Indeed, take a presheaf X over $\mathbb{D}$, Δ_φ is the natural transformation from XVT to XUS given at a by $(\Delta_\varphi)_a \colon X(V(T(a))) \xrightarrow{X(\varphi_a)} X(U(S(a)))$. This new square gives by adjunction (the so-called mate calculus) two further squares as on the left and right:

$$\begin{tikzcd}
\widehat{\mathbb{A}} \arrow[d, "\Sigma_S"'] & \widehat{\mathbb{B}} \arrow[l, "\Delta_T"'] \arrow[d, "\Sigma_V"] \arrow[dl, phantom, "\Downarrow\Sigma_\varphi"] \\
\widehat{\mathbb{C}} & \widehat{\mathbb{D}} \arrow[l, "\Delta_U"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\widehat{\mathbb{A}} \arrow[r, "\Pi_T"] & \widehat{\mathbb{B}} \\
\widehat{\mathbb{C}} \arrow[u, "\Delta_S"] \arrow[r, "\Pi_U"'] \arrow[ur, phantom, "\overset{\Pi_\varphi}{\Leftarrow}"] & \widehat{\mathbb{D}}. \arrow[u, "\Delta_V"']
\end{tikzcd}$$

For example, the left-hand one is defined as

$$\begin{tikzcd}
 & \widehat{\mathbb{A}} \arrow[dl, "\Sigma_S"'] & \widehat{\mathbb{B}} \arrow[l, "\Delta_T"'] \arrow[r, equal] & \widehat{\mathbb{B}} \arrow[dl, "\Sigma_V"] \\
\widehat{\mathbb{C}} \arrow[r, equal] \arrow[ur, phantom, "\varepsilon^S\Downarrow"] & \widehat{\mathbb{C}} \arrow[u, "\Delta_S"] \arrow[ur, phantom, "\Delta_\varphi\swarrow"] & \widehat{\mathbb{D}}, \arrow[l, "\Delta_U"] \arrow[u, "\Delta_V"'] \arrow[ur, phantom, "\Downarrow\eta^V"] &
\end{tikzcd}$$

where η^S and ε^V come from the adjunctions $\sum_S \dashv \Delta_S$ and $\sum_V \dashv \Delta_V$ respectively, and the right-hand square is defined similarly using the adjunctions $\Delta_T \dashv \prod_T$ and $\Delta_U \dashv \prod_U$.

Definition 2.2.61. *A square φ is* exact *if and only if $\sum_\varphi$ is an isomorphism.*

The following is well-known:

Property 2.2.62. *A square φ is exact if and only if $\prod_\varphi$ is an isomorphism.*

The notion of exactness thus corresponds to commutation up to isomorphism of the two diagrams above.

Remark. *Our experience with exact squares suggests that it may be difficult to remember the directions of $\sum$'s, Δ's and $\prod$'s, as well as of the induced natural transformations. In an attempt to try and spare the reader a few headaches, our mnemonic is that the original transformation φ points* to *the $\sum$'ed functor if we are to reason about $\sum_\varphi$, and* from *the $\prod$'ed functor if we are to reason about $\prod_\varphi$. Furthermore, induced natural transformations flow along left adjoints and against right adjoints (hence along $\sum$'s for $\sum_\varphi$, and against $\prod$'s for $\prod_\varphi$).*

Notation 2.2.63. *In the following, we will manipulate diagrams of restriction and extension functors as above. In order to reduce some notational clutter, hats will be omitted and arrows will point in the direction of underlying functors. This means that Δ arrows will point in the "wrong" direction, in the sense that if $F: X \to Y$, we will write $Y \xleftarrow{\Delta_F} X$ for $\widehat{Y} \xrightarrow{\Delta_F} \widehat{X}$. We find that this notation also spares us the mental overhead of wondering about the direction of underlying functors and matching over $\prod, \sum$ and Δ. The price to pay is that when we draw commuting diagrams, sources and sinks are sometimes less easy to spot. We try and tame this difficulty by writing our diagrams from left to right (and from top to bottom for vertical arrows) when possible.*

Lemma 2.2.64 (Guitart [45, Theorem 1.2] (zigzag criterion))**.** *A square as in (2.2) is exact if and only if, for all objects c of $\mathbb{C}$, b of $\mathbb{B}$, and morphisms $f: Uc \to Vb$:*

- *there is an object a of $\mathbb{A}$ such that f factors as $Uc \xrightarrow{Ug} USa \xrightarrow{\varphi_a} VTa \xrightarrow{Vh} Vb$,*

- *and for all two such factorisations* $Uc \xrightarrow{Ug} USa \xrightarrow{\varphi_a} VTa \xrightarrow{Vh} Vb$ *and* $Uc \xrightarrow{Ug'} USa' \xrightarrow{\varphi_{a'}} VTa' \xrightarrow{Vh'} Vb$*, there exists a commuting diagram (which Guitart calls a* lantern*)*

$$\begin{tikzcd}
 & & c \arrow[dll, "g"'] \arrow[dl] \arrow[dr] \arrow[drr, "g'"] & & \\
Sa \arrow[r, "St_0"'] & \cdot & \cdots \arrow[l, "St_1"] \arrow[r, "St_{n-1}"'] & \cdot & Sa' \arrow[l, "St_n"] \\
Ta \arrow[r, "Tt_0"] \arrow[drr, "h"'] & \cdot \arrow[dr] & \cdots \arrow[l, "Tt_1"'] \arrow[r, "Tt_{n-1}"] & \cdot \arrow[dl] & Ta' \arrow[l, "Tt_n"'] \arrow[dll, "h'"] \\
 & & b. & &
\end{tikzcd}$$

Here is a list of useful examples of exact squares:

Lemma 2.2.65 (Guitart [45, Examples 1.14.2 and 1.14.3])**.** *Any comma (resp. cocomma) square is exact.*

Lemma 2.2.66 (Guitart [45, Examples 1.14.1 and 1.14.4])**.** *For any functor* $f\colon A \to B$*, the square below left is exact; furthermore, the square below right is exact if and only if* f *is fully faithful:*

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "f"] \arrow[d, equal] \arrow[dr, phantom, "\overset{id_f}{\Rightarrow}"] & \mathbb{B} \arrow[d, equal] \\
\mathbb{A} \arrow[r, "f"'] & \mathbb{B}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathbb{A} \arrow[r, equal] \arrow[d, equal] \arrow[dr, phantom, "\overset{id_f}{\Rightarrow}"] & \mathbb{A} \arrow[d, "f"] \\
\mathbb{A} \arrow[r, "f"'] & \mathbb{B}
\end{tikzcd}$$

Lemma 2.2.67 (Guitart [45, Theorem 1.8])**.** *Exact squares are stable under horizontal composition.*

Lemma 2.2.68. *Any square* (2.2) *in which* φ *and* S *are identities and* V *is fully faithful is exact.*

Proof. We obtain the given square as the horizontal composite

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "T"'] \arrow[d, equal] \arrow[rr, bend left, "T"] & \mathbb{B} \arrow[r, equal] \arrow[d, equal] & \mathbb{B} \arrow[d, "V"] \\
\mathbb{A} \arrow[r, "T"] \arrow[rr, bend right, "U"'] & \mathbb{B} \arrow[r, "V"] & \mathbb{C}.
\end{tikzcd}$$

The big square is exact by Lemma 2.2.67 because it is the horizontal composite of two squares that are exact by Lemma 2.2.66. □

A slightly more general version of this result is:

Lemma 2.2.69. *Any square as in* (2.2) *in which* φ *is an identity,* S *is an equivalence, and* V *is fully faithful is exact.*

Proof. By the zigzag criterion. Consider a morphism $f: Uc \to Vb$. Since S is an equivalence, there exists a functor $S': \mathbb{C} \to \mathbb{A}$ and a natural isomorphism $\eta: \mathrm{Id}_{\mathbb{C}} \to SS'$. We can thus factor f as $Uc \xrightarrow{U\eta_c} USS'c = VTS'c \xrightarrow{f\circ(U\eta_c^{-1})} Vb$. Since V is full, $f \circ (U\eta_c^{-1})$ is the image of some h through V, which gives us the first point of the criterion.

Now, assume a decomposition of f as $Uc \xrightarrow{Ug'} USa = VTa \xrightarrow{Vh'} Vb$. Because S is an equivalence, it is in particular full, so $g'\eta_c^{-1}$ is the image of some k through S. Our candidate lantern is then:

$$\begin{tikzcd}
& c \arrow[dl, "\eta_c"'] \arrow[dr, "g'"] & \\
SS'c \arrow[rr, "Sk"'] & & Sa \\
TS'c \arrow[rr, "Tk"] \arrow[dr, "h"'] & & Ta \arrow[dl, "h'"] \\
& b. &
\end{tikzcd}$$

The top part commutes because $Sk = g'\eta_c^{-1}$ by definition of k, and the bottom part commute because it does when V is applied to it, and V is faithful. $\square$

Here are two lemmas showing that pullbacks of (op)fibrations are exact squares (though not in the same direction!):

Lemma 2.2.70. *Any pullback square* (2.2) *with* V *a fibration is exact.*

Proof. We proceed by the zigzag criterion. Consider any $f: Uc \to Vb$. Let us first establish existence of the desired factorisation, by considering any cartesian lifting $l: b_0 \to b$ of b along f. We obtain $Uc = Vb_0$, hence by universal property of pullback a unique a such that $c = Sa$ and $b_0 = Ta$. The original f thus factors as

$$Uc \xrightarrow{U(id)} USa \xrightarrow{id} VTa \xrightarrow{Vl} Vb.$$

We now need to show that any factorisation is connected to this one by some lantern. So consider any factorisation of f as

$$Uc \xrightarrow{Ug} USa' \xrightarrow{id} VTa' \xrightarrow{Vh} Vb.$$

Because fibrations are stable under pullback, S is a fibration, so we may pick a cartesian lifting, say $k: a'' \to a'$, of a' along g, so that in particular $Sa'' = c$ and $Sk = g$. Now, we have $VTa'' = USa'' = Uc = USa = VTa$, hence a commuting square as the bottom part of

$$\begin{tikzcd}
Ta'' \arrow[rr, "Tk"] \arrow[dr, dashed, "u"'] & & Ta' \arrow[dr, "h"] & \\
& Ta \arrow[rr, "l"'] & & b
\end{tikzcd}$$

$$\begin{tikzcd}
VTa'' \arrow[rr, "VTk"] \arrow[dr, equal] & & VTa' \arrow[dr, "Vh"] & \\
& VTa \arrow[rr, "Vb"'] & & Vb
\end{tikzcd}$$

(because $VTk = USk = Ug$ so $Vh \circ VTk = f = Vl$). So by cartesianness of l, we obtain a dashed u making the top square above commute. But since $Vu = id$, we have that $U(id) = Vu$ so by universal property of pullback again there exists a unique $w{:}\, a'' \to a$ such that $Sw = id$ and $Tw = u$. We thus obtain the following lantern:

$$\begin{tikzcd}
& c \arrow[dl, "g"'] \arrow[d, equal] \arrow[dr, equal] & \\
Sa' & Sa'' \arrow[l, "Sk"] \arrow[r, equal, "Sw"'] & Sa
\end{tikzcd}$$

$$\begin{tikzcd}
Ta' \arrow[dr, "h"'] & Ta'' \arrow[l, "Tk"'] \arrow[r, "Tw"] \arrow[d] & Ta \arrow[dl, "l"] \\
& b, &
\end{tikzcd}$$

as desired. □

Lemma 2.2.71. *Any pullback square* (2.2) *with U an opfibration is exact.*

Proof. By [45, Theorem 1.9], φ is exact if and only if φ^{op} is exact. So, if φ is a pullback and U is an opfibration, then φ^{op} is also a pullback, and its "right-hand" morphism is U^{op}, which is a fibration, so φ^{op} is exact by Lemma 2.2.70, and thus so is φ. □

2.2.7 Sheaves

Required:	2.2.5.
Recommended:	2.2.1,2.2.4.

While not used as extensively as presheaves, and hidden much deeper in the details, sheaves are also an essential structure that appears throughout our work. They are at the heart of the recasting of innocence in game semantics by Hirschowitz [50], later reused by Tsukada and Ong [97].

A presheaf X on $\mathbb{C}$ is a sheaf when it has unique *amalgamations*, which basically means that, if we know the value of X on sufficiently many subobjects f_i of c, then we can uniquely recover the value of X at c (we say that the f_i's *cover* c). The intuition comes from topological spaces: if we take $\mathbb{C}$ to be the category of open sets of some topological space (with inclusions as morphisms), then we can uniquely recover the value of a function f on an open set U if we are given its restrictions to open sets V_i that cover U. Our presentation is largely based on Mac Lane and Moerdijk [77] and Johnstone [57].

Our running example will be that of the category of open sets of a topological space X:

Example 2.2.72. *If X is a topological space, we may define $\mathbb{C}$ as the category whose objects are open sets of X and whose morphisms are inclusions thereof. We can then define the set of functions from X to $\mathbb{R}$ as a presheaf P on $\mathbb{C}$ whose sets $P(U)$ are $\mathbb{R}^U$, the sets of all functions from U to $\mathbb{R}$, and whose action on morphisms is restriction. We can similarly define the set of continuous or bounded functions from X to $\mathbb{R}$.*

Let us take an open set U of X and a family of open sets V_i such that the union $\bigcup_i V_i = U$, and assume we are given functions f_i for all V_i that are compatible in the sense that $f_{i|V_i \cap V_j} = f_{j|V_i \cap V_j}$. Then, there is a single way to define a function f on U whose restriction to each V_i is f_i. Moreover, if all f_i's are continuous, then so is f. What this means is that we can amalgamate *the f_i's into a single f, and we say that the presheaf of functions (or continuous functions) is a* sheaf. *However, even if all f_i's are bounded, f may not be (take $U = \mathbb{R}$, $V_i = (-i, i)$, and $f_i = id_{V_i}$), so the presheaf of bounded functions is not a sheaf: we cannot amalgamate the f_i's into an f.*

Definition 2.2.73. *Let $\mathbb{C}$ be a small category and c an object of $\mathbb{C}$. A* sieve *S on c is a family of morphisms $\{f_i : c_i \to c\}$ that is closed under pre-composition, i.e., if $f : c' \to c$ is in S and $g : c'' \to c'$ is a morphism, then fg is also in S.*

Note that a sieve S on c is actually a subpresheaf of y_c defined on c' by the set of all $f : c' \to c$ in S and whose action on morphisms is pre-composition.

Example 2.2.74. *If $\mathbb{C}$ is the category of open sets of a topological space X ordered by inclusion, then a sieve on U is a downwards-closed family of open sets $V_i \subseteq U$.*

Given a sieve S on c and a morphism $f : c' \to c$, one may define the pullback $f^*(S)$ of S along f as the family of morphisms g such that fg is in S.

Property 2.2.75. *For all objects c, sieves S on c and morphisms $f : c' \to c$, $f^*(S)$ is a sieve.*

Example 2.2.76. *In the case of open sets of a topological space, if $S = \{V_i\}$ is a sieve on U and V is included in U, then the construction above gives $\{V_i \cap V\}$.*

Definition 2.2.77. *A* Grothendieck topology *(or simply a* topology*) J on a category $\mathbb{C}$ is a functor that assigns to each object c of $\mathbb{C}$ a collection of sieves $J(c)$ which are called J-*covering *(or simply* covering*), and acts on morphisms by pullback. Covering sieves must enjoy the following properties:*

- *the maximal sieve of all maps to c must be in $J(c)$,*
- *if S is in $J(c)$ and $f : c' \to c$, then $f^*(S)$ is in $J(c')$,*
- *if S is in $J(c)$ and R is a sieve on c such that, for all $f : c' \to c$ in S, $f^*(R)$ is in $J(c')$, then R is in $J(c)$.*

These conditions are actually very natural. If we think of $f : c' \to c$ as "picking" a piece of c (which is exactly what happens with topological spaces), then the conditions above can be rephrased in words as:

- taking all pieces of c must cover c,
- if some pieces of c cover c, then restricting these pieces to c' must cover c',
- if some pieces c_i cover c and we consider some other pieces c'_j, then the only way for c'_j not to cover c is for them not to cover one of the c_i's.

Example 2.2.78. *Some simple examples of topologies are:*

- *the trivial topology, where the only sieve that covers c is the sieve $\{f: c' \to c\}$ of all morphisms to c,*
- *the coarsest topology, where any sieve S on c covers it,*
- *given a topological space X and $\mathbb{C}$ defined as the category of open sets of X, a sieve $\{V_i\}$ covers U if $\bigcup_i V_i = U$.*

An example of crucial importance in this dissertation is that of *topology induced by a functor*, which can mean two different things. The first one is, given a category $\mathbb{D}$ equipped with a topology J, a way to transfer the topology on J to a subcategory of $\mathbb{D}$. The other one, which we are interested in here, is, given a functor $F: \mathbb{C} \to \mathbb{D}$, a way to equip $\mathbb{D}$ with a topology basically saying that an object d is covered by the collection of objects from c that map into d.

Example 2.2.79. *If $F: \mathbb{C} \to \mathbb{D}$ is a functor, then a sieve S is covering for the topology J induced by F if it contains all morphisms of the form $Fc \to d$. This is indeed a topology because:*

- *the sieve of all morphisms to d indeed contains all morphisms $Fc \to d$,*
- *if S contains all morphisms $Fc \to d$ and $f: d' \to d$ is a morphism, then for all $g: Fc \to d'$, fg is a map from Fc to d, so it is in S, and therefore g is in $f^*(S)$, which is thus a covering sieve,*
- *if S and R verify the hypotheses of the third item in the definition above, then, in particular, for all $f: Fc \to d$, $f^*(R)$ contains all morphisms $Fc' \to Fc$, so in particular it contains $F(id_c) = id_{Fc}$, so $f = f id_{Fc}$ is in R, and so R indeed covers c.*

Definition 2.2.80. *Let $\mathbb{C}$ be a small category equipped with a topology J, and S be a covering sieve on c. If X is a presheaf over $\mathbb{C}$, an S-*matching family *of elements of X assigns to each element $f: c' \to c$ of S an element x_f of $X(c')$ such that $x_f \cdot g = x_{fg}$ for all $f: c' \to c$ in S and $g: c'' \to c'$.*

An amalgamation *of such a matching family is an element x in $X(c)$ such that $x \cdot f = x_f$ for all f in S.*

Note that an S-matching family is equivalently a natural transformation from S (seen as a subpresheaf of y_c) to X. An amalgamation is thus a morphism $\mathsf{y}_c \to X$ such that

$$\begin{tikzcd}
S \arrow[r, hook] \arrow[d] & \mathsf{y}_c \arrow[dl] \\
X &
\end{tikzcd}$$

commutes.

Example 2.2.81. *If S is the maximal sieve of all morphisms to c, an S-matching family of elements of X is a family x_f indexed by all morphisms $f\colon c' \to c$ of elements of $X(c')$, such that $x_f \cdot g = x_{fg}$. An amalgamation of such a family is an element x of $X(c)$ such that $x \cdot f = x_f$ for all f. But then, necessarily $x = x \cdot id_c = x_{id_c}$, and such a choice indeed works, because $x \cdot f = x_{id_c} \cdot f = x_f$.*

If the empty sieve covers c, a matching family of elements of X for it is an empty family. An amalgamation of such a family is any element in $X(c)$.

Let $\mathbb{C}$ be the category of open sets of a topological space X equipped with the topology generated by these open sets and P is a presheaf over $\mathbb{C}$. Given an open set U, a sieve $S = \{V_i\}$ that covers U, and an element x in $P(V_i)$ we denote by $x_{|V_j}$ the "restriction" of x to V_j, defined as $x \cdot f$, where f is the inclusion of V_j into V_i. An S-matching family of elements of P for $\{V_i\}$ is then a family of elements x_i of $P(V_i)$ such that $x_{i|V_i \cap V_j} = x_{j|V_i \cap V_j}$ for all V_i and V_j. Indeed, if $V_k = V_i \cap V_j$, we have that $x_{i|V_i \cap V_j} = x_k = x_{j|V_i \cap V_j}$. An amalgamation of such a family is an x in $P(U)$ such that $x_{|V_i} = x_i$ for all V_i.

In particular, if P is the presheaf of functions to $\mathbb{R}$, matching families and amalgamations are exactly what is discussed in Example 2.2.72.

Definition 2.2.82. *If $\mathbb{C}$ is a small category and P is a presheaf on $\mathbb{C}$, P is a* sheaf *for the topology J if, for all objects c and sieves S covering c, there is a unique amalgamation to each S-matching family. We denote by* $\mathrm{Sh}(\mathbb{D}, J)$ *the full subcategory of $\widehat{\mathbb{D}}$ spanning sheaves for J (or* $\mathrm{Sh}(\mathbb{D})$ *if J is clear from context).*

If we see sieves as subpresheaves of some y_c, P is a sheaf when, for all objects c and sieves S covering c, the restriction map $Y(c) \cong [\mathsf{y}_c, Y] \to [S, Y]$ is a bijection.

Example 2.2.83. *We have already seen that, in the case of the trivial topology, no matter the presheaf, there is always a unique amalgamation to any matching family, so all presheaves are sheaves.*

In the case of the coarsest topology, there must be a unique amalgamation for any matching family of any covering sieve, in particular the empty sieve, so $X(c)$ must be a singleton, and sheaves are exactly terminal presheaves.

In the case of topological spaces, the presheaves of functions and that of continuous functions are sheaves, but that of bounded functions is not.

Finally, in the case of the topology induced by a functor, which is the case we are interested in, we have the following characterisation:

Lemma 2.2.84. *If $F\colon \mathbb{C} \to \mathbb{D}$ is fully faithful and Y is a presheaf over $\mathbb{D}$, then Y is a sheaf if and only if it is in the essential image of $\prod_F$.*

Proof. Let J_F denote the topology induced by F. Further let $i\colon \mathrm{Sh}(\mathbb{D}, J_F) \hookrightarrow \widehat{\mathbb{D}}$ denote the embedding of sheaves into presheaves.

F is cover-reflecting (if we consider the trivial topology on $\mathbb{C}$), which means that covering sieves in $\mathbb{D}$ of objects in $\mathbb{C}$ can be transported back to covering sieves in $\mathbb{C}$. Formally, a functor F is cover-reflecting if, for all objects c of $\mathbb{C}$

and sieve R covering Fc, there is a sieve S covering c such that $F(f)$ is in R for all morphisms f in S. In our case, this is obvious because a sieve R covers Fc if and only if it contains F/Fc, so we may take S to be the full covering sieve.

By [57, Proposition C 2.3.18], $\prod_F$ thus factors through i, say as R, because all presheaves over $\mathbb{C}$ are sheaves. Therefore, the adjunction $\Delta_F \dashv \prod_F$ restricts to an adjunction $(\Delta_F \circ i) \dashv R$ because i is fully faithful. This is an adjoint equivalence by the comparison lemma [57, Lemma C 2.2.3]. So $\widehat{\mathbb{C}}$ and $\mathrm{Sh}(\mathbb{D}, J_F)$ are equivalent through $\prod_F$, hence the result. □

2.2.8 Factorisation Systems

Required: ∅.
Recommended: ∅.

Factorisation systems are a tool we will use extensively throughout this dissertation. The property that gives factorisation systems their name is that they can factor any arrow of a given category as a composite of arrows of two given classes $\mathcal{L}$ and $\mathcal{R}$. Another property of factorisation systems is that $\mathcal{L}$ and $\mathcal{R}$ are *orthogonal*, which ensures the existence of diagonal fillers for particular commuting squares.

In any category $\mathcal{C}$, we say that $l\colon A \to C$ is *left orthogonal* to $r\colon B \to D$, or equivalently that r is *right orthogonal* to l, if and only if for all commuting squares as the solid part of

$$\begin{tikzcd} A \arrow[r, "u"] \arrow[d, "l"'] & B \arrow[d, "r"] \\ C \arrow[r, "v"'] \arrow[ur, dashed, "d"] & D, \end{tikzcd}$$

there exists a unique d as shown making both triangles commute.

Notation 2.2.85. *We denote by $l \perp r$ the existence of a unique such d for all u and v, and extend the notation to sets of arrows by writing $\mathcal{L} \perp \mathcal{R}$ when $l \perp r$ for all $l \in \mathcal{L}$ and $r \in \mathcal{R}$. Similarly, $\mathcal{L}^{\perp}$ denotes the class of all arrows that are right orthogonal to all arrows of $\mathcal{L}$, and symmetrically for ${}^{\perp}\mathcal{R}$.*

Definition 2.2.86. *A* factorisation system *[17] on a category $\mathcal{C}$ consists of two classes of maps $\mathcal{L}$ and $\mathcal{R}$ such that $\mathcal{L} = {}^{\perp}\mathcal{R}$, $\mathcal{L}^{\perp} = \mathcal{R}$, and any morphism $f\colon C \to D$ factors as $C \xrightarrow{l_f} A_f \xrightarrow{r_f} D$ with $l_f \in \mathcal{L}$ and $r_f \in \mathcal{R}$.*

Example 2.2.87. *The first example of a factorisation system is given by the classes* Epi *and* Mono*, respectively of surjections and injections, in* Set. *Indeed, it is well-known that all functions factor through their images as a surjection followed by an injection. Perhaps less well-known is that* Epi *and* Mono *are orthogonal. Indeed, take a commuting square*

$$\begin{tikzcd} X \arrow[r, "u"] \arrow[d, "l"'] & X' \arrow[d, "r"] \\ Y \arrow[r, "v"'] & Y' \end{tikzcd}$$

with $l \in \mathrm{Epi}$ and $r \in \mathrm{Mono}$. Then the definition $d(l(x)) = u(x)$ indeed defines a function: any y in Y has an image through d because l is surjective, and the definition does not depend on the chosen x because, if $l(x_1) = l(x_2)$, then $r(d(l(x_1))) = r(u(x_1)) = v(l(x_1)) = v(l(x_2)) = r(u(x_2)) = r(d(l(x_2)))$, and r is injective. This is the only choice of function from Y to X' that makes both triangles commute because r is mono.

This extends to presheaf categories: for any small category $\mathbb{C}$, epi and monic natural transformations form a factorisation system on $\widehat{\mathbb{C}}$, which we also denote by $(\mathrm{Epi}, \mathrm{Mono})$. The factorisation of arrows is the same as in the previous case, the only difference is that we need to show that the definition of the image is a presheaf, and that the transformations going to and coming from the image are natural, which is easy. For orthogonality, if we are in the same case as above, but where objects of the diagram are presheaves and arrows are natural transformations, then the definition $d_c(l_c(x)) = u_c(x)$ is the only choice of function that makes both triangles commute, by the same arguments as above, and because a natural transformation is epi (or mono) exactly when all its components are. It then remains to show that this mapping is natural, i.e. that, for all $f{:}\, c \to c'$, $d_c(y \cdot f) = d_{c'}(y) \cdot f$, but that is easy because if $y = l_{c'}(x)$, then $d_{c'}(y) \cdot f = d_{c'}(l_{c'}(x)) \cdot f = u_{c'}(x) \cdot f = u_c(x \cdot f) = d_c(l_c(x \cdot f)) = d_c(l_c(x) \cdot f) = d_c(y \cdot f)$.

Another example, in Cat *this time, is $(\mathrm{Bo}, \mathrm{FF})$, where Bo is the class of functors that are bijective on objects and FF is the class of fully faithful functors. All functors $F{:}\, \mathbb{C} \to \mathbb{D}$ factor through $\mathbb{E}$, which has the same objects as $\mathbb{C}$, and whose homsets are defined by $\mathbb{E}(c, c') = \mathbb{D}(Fc, Fc')$. The functor $\mathbb{C} \to \mathbb{E}$ is the identity on objects but sends f to Ff and the functor $\mathbb{E} \to \mathbb{D}$ is the identity on morphisms but maps c to Fc. When faced with a commuting square*

$$\begin{tikzcd}
C \arrow[r, "U"] \arrow[d, "L"'] & C' \arrow[d, "R"] \\
D \arrow[r, "V"'] & D',
\end{tikzcd}$$

where L is bijective on objects and R is fully faithful, the diagonal filler $\Delta{:}\, D \to C'$ may be defined as:

- *$\Delta(d) = U(c)$, where c is the only object of C such that $L(c) = d$ and*
- *$\Delta(f{:}\, d_1 \to d_2) = g$, where g is defined as follows: if we call c_i the only object of C such that $L(c_i) = d_i$, then $R(U(c_i)) = V(L(c_i)) = Vd_i$, so by full faithfulness of R, Vf has a unique antecedent $g{:}\, c_1 \to c_2$.*

It is then easy to show that this choice of Δ is the only one that makes both triangles commute.

The following characterisation of factorisation systems sometimes gives a better intuition:

Property 2.2.88. *A pair $(\mathcal{L}, \mathcal{R})$ of classes of maps of $\mathcal{C}$ form a factorisation system exactly when:*

- *all arrows $f{:}\, C \to D$ may be factored as $C \xrightarrow{l_f} A_f \xrightarrow{r_f} D$, with $l_f \in \mathcal{L}$ and $r_f \in \mathcal{R}$* and that factorisation is unique up to a unique isomorphism *and*

- *$\mathcal{L}$ and $\mathcal{R}$ contain the isomorphisms and are closed under composition.*

Using this property, we can easily give a non-example:

Example 2.2.89. *The classes* (Mono, Epi) *do not form a factorisation system on* Set*. Indeed, all arrows factor as an injection followed by a surjection, but that factorisation may not be unique. Consider any function $f\colon X \to Y$, then f may be factored through its graph as $X \xrightarrow{l} X \times Y \xrightarrow{r} Y$, with*

$$l\colon \left| \begin{array}{ccc} X & \to & X \times Y \\ x & \mapsto & (x, f(x)) \end{array} \right. \quad \textit{and } r\colon \left| \begin{array}{ccc} X \times Y & \to & Y \\ (x, y) & \mapsto & y, \end{array} \right.$$

but it may also be factored through its (generally not isomorphic) cograph as $X \xrightarrow{l} X + Y \xrightarrow{r} Y$, with

$$l\colon \left| \begin{array}{ccc} X & \to & X + Y \\ x & \mapsto & \operatorname{inl} x \end{array} \right. \quad \textit{and } r\colon \left| \begin{array}{ccc} X + Y & \to & Y \\ \operatorname{inl} x & \mapsto & f(x) \\ \operatorname{inr} y & \mapsto & y \end{array} \right. \quad \textit{or}$$

$$l\colon \left| \begin{array}{ccc} X & \to & X + Y \\ x & \mapsto & \operatorname{inr} f(x) \end{array} \right. \quad \textit{and the same } r.$$

In terms of orthogonality, the classes Mono *and* Epi *are* weakly orthogonal*, in the sense that for all commuting squares*

$$\begin{tikzcd}
A \arrow[r, "u"] \arrow[d, "l"'] & B \arrow[d, "r"] \\
C \arrow[r, "v"'] & D
\end{tikzcd}$$

with l an injection and r a surjection, there exists a diagonal filler, but it is not necessarily unique. For example,

$$\begin{tikzcd}
0 \arrow[r] \arrow[d] & 2 \arrow[d] \\
1 \arrow[r] & 1
\end{tikzcd}$$

possesses two distinct diagonal fillers.

The factorisation systems that we will be interested in will be *cofibrantly generated*. *Cofibrant generation* refers to the fact that $\mathcal{L}$ and $\mathcal{R}$ are defined from some generating set J, merely by the lifting property: $\mathcal{R} = J^{\perp}$ and $\mathcal{L} = {}^{\perp}\mathcal{R}$. The point here is that J is a set, rather than a class. In fact, in many useful cases, it is even a rather small set. In our case, it will be bounded by the cardinality of $\mathbb{C}$. Though it is not trivial – this uses the famous "small object" argument, we have:

Theorem 2.2.90 (Bousfield [17])**.** *For any set J of maps in any locally presentable category $\mathbb{C}$, $({}^{\perp}(J^{\perp}), J^{\perp})$ forms a factorisation system. We say this factorisation system is* cofibrantly generated by *J.*

Example 2.2.91. *The* (Epi, Mono) *factorisation system on* Set *is cofibrantly generated by the singleton* $\{2 \to 1\}$*. For any* $\mathbb{C}$*, the* (Epi, Mono) *factorisation system on* $\widehat{\mathbb{C}}$ *is cofibrantly generated by the set of all maps* $\nabla_{\mathsf{y}_c} \colon \mathsf{y}_c + \mathsf{y}_c \to \mathsf{y}_c$*, for* $c \in \mathrm{ob}(\mathbb{C})$*.*

A crucial property of factorisation systems is the following:

Lemma 2.2.92. *For any factorisation system* $(\mathcal{L}, \mathcal{R})$*,* $\mathcal{L}$ *contains all isomorphisms and is stable under right cancellation, composition and pushout.*

Stability under right cancellation means that if some composite $g \circ h$ *is in* $\mathcal{L}$ *for* $h \in \mathcal{L}$*, then so is* g*.*

Stability under pushout means that given any pushout square

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "l"'] & B \arrow[d, "l'"] \\
C \arrow[r, "g"'] & D, \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

if $l \in \mathcal{L}$*, then also* $l' \in \mathcal{L}$*.*

Dually, $\mathcal{R}$ *contains all isomorphisms and is stable under left cancellation, composition and pullback (in the obvious dual sense to stability under right cancellation and pushout).*

So in particular both classes determine identity-on-objects subcategories of $\mathcal{C}$.

2.2.9 Pseudo Double Categories

Required: ∅.
Recommended: ∅.

A *pseudo double category* [31, 32, 42, 43, 72, 37] $\mathbb{D}$ consists of a set $\mathrm{ob}(\mathbb{D})$ of *objects*, shared by a "horizontal" category $\mathbb{D}_h$ and a "vertical" bicategory $\mathbb{D}_v$. Following Paré [90], $\mathbb{D}_h$, merely being a category, has standard notation (normal arrows, $\circ$ for composition, *id* for identities), while the bicategory $\mathbb{D}_v$ earns fancier notation ($\xrightarrow{\bullet}$ for arrows, $\bullet$ for composition, $id^\bullet$ for identities). Moreover, for each "square" as on the left below, $\mathbb{D}$ comes equipped with a set of *cells* α, which have vertical, resp. horizontal, domains and codomains, denoted by $\mathrm{dom}_v(\alpha)$, $\mathrm{cod}_v(\alpha)$, $\mathrm{dom}_h(\alpha)$, and $\mathrm{cod}_h(\alpha)$. We picture this as on the right below.

$$\begin{tikzcd}
X' \arrow[r, "h"] \arrow[d, "u"', "\bullet" marking] & Y' \arrow[d, "u'", "\bullet" marking] \\
X \arrow[r, "h'"'] & Y
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
X' \arrow[r, "h"] \arrow[d, "u"', "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha}{\Longrightarrow}"] & Y' \arrow[d, "u'", "\bullet" marking] \\
X \arrow[r, "h'"'] & Y
\end{tikzcd}$$

where $u = \mathrm{dom}_h(\alpha)$, $u' = \mathrm{cod}_h(\alpha)$, $h = \mathrm{dom}_v(\alpha)$, and $h' = \mathrm{cod}_v(\alpha)$. $\mathbb{D}$ is furthermore equipped with operations for composing cells: $\circ$ composes them along a common vertical morphism, $\bullet$ composes along horizontal morphisms. Both vertical compositions (of morphisms and cells) may only be associative up to coherent isomorphism. The full axiomatisation is given by Garner [37], and we here only mention the *interchange law*, which says that both ways of parsing the following diagram coincide:

$$\begin{tikzcd}
X \arrow[r, "l"] \arrow[d, "u"', "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha}{\Longrightarrow}"] & X' \arrow[r, "l'"] \arrow[d, "u'", "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha'}{\Longrightarrow}"] & X'' \arrow[d, "u''", "\bullet" marking] \\
Y \arrow[r, "k"] \arrow[d, "v"', "\bullet" marking] \arrow[dr, phantom, "\overset{\beta}{\Longrightarrow}"] & Y' \arrow[r, "k'"] \arrow[d, "v'", "\bullet" marking] \arrow[dr, phantom, "\overset{\beta'}{\Longrightarrow}"] & Y'' \arrow[d, "v''", "\bullet" marking] \\
Z \arrow[r, "h"'] & Z' \arrow[r, "h'"'] & Z'',
\end{tikzcd}$$

i.e., $(\beta' \circ \beta) \bullet (\alpha' \circ \alpha) = (\beta' \bullet \alpha') \circ (\beta \bullet \alpha)$.

For any pseudo double category $\mathbb{D}$, we denote by $\mathbb{D}_H$ the category with vertical morphisms as objects and cells as morphisms, and by $\mathbb{D}_V$ the bicategory with horizontal morphisms as objects and cells as morphisms. Domain and codomain maps extend to functors $\mathrm{dom}_v, \mathrm{cod}_v \colon \mathbb{D}_H \to \mathbb{D}_h$ and $\mathrm{dom}_h, \mathrm{cod}_h \colon \mathbb{D}_V \to \mathbb{D}_v$. We will refer to dom_v and cod_v simply as dom and cod, reserving subscripts for dom_h and cod_h.

Some of our constructions will be based on this example:

Example 2.2.93. *Starting from any category $\mathcal{C}$ with pushouts, consider the pseudo double category $\mathsf{Cospan}(\mathcal{C})$ with*

- *$\mathcal{C}$ itself as horizontal category, i.e., $\mathsf{Cospan}(\mathcal{C})_h = \mathcal{C}$,*
- *as vertical morphisms $X \mathrel{\bullet\!\!\!\to} Y$ all cospans $X \to U \leftarrow Y$, and*
- *as cells all commuting diagrams*

$$\begin{tikzcd}
X \arrow[r, "k"] \arrow[d] & X' \arrow[d] \\
U \arrow[r, "l"] & U' \\
Y \arrow[r, "h"'] \arrow[u] & Y', \arrow[u]
\end{tikzcd} \tag{2.3}$$

with $\mathrm{dom}(k, l, h) = k$, $\mathrm{dom}_h(k, l, h) = (X \to U \leftarrow Y)$, *etc.*

Composition in $\mathsf{Cospan}(\mathcal{C})_v$ is defined by (some global choice of) pushout and composition in $\mathsf{Cospan}(\mathcal{C})_V$ follows by universal property of pushout.

Chapter 3

Presheaves and Concurrent Traces

3.1 Introduction

In this chapter, we give concrete examples of how to represent concurrent traces of executions as presheaves. We give three different applications: Petri nets, graph rewriting, and interaction nets [65]. We do not consider the results discussed in this chapter to be very important, as it is mostly meant to be an introduction to the use we will make of presheaves in Chapters 4 and 5 to model concurrent execution traces. However, we do have a nice result about how to represent the unfolding of a system, which is significantly simpler than previous presentations.

This work is inspired by work by Baldan et al. [10], who give a general framework for rewriting systems based on adhesive categories. They give a definition of *process* in their setting, which is a sequence of rewriting steps considered up to permutation of independent steps, and they show how to build the *unfolding* of any object. Here, we propose a different approach to model rewriting, processes, and unfoldings, which is based on string diagrams. The main contribution of our work is a direct, combinatorial description of processes and unfoldings in terms of presheaves. Just like in Baldan et al.'s work, the unfolding of an object has a universal property (slightly different from theirs, though very close in spirit). This framework can adequately model executions of Petri nets or interaction nets [65], as well as processes in simple concurrent languages such as CCS [50], the π-calculus [29], or the join-calculus [36]. It can also model some form of graph rewriting, which has yet to be compared to existing frameworks, such as DPO [33] and SPO [34]. Finally, the same techniques can model interaction in HON game semantics [25] (which is described in Chapter 5).

The main difficulty in Baldan et al.'s work is that a process is more than the sequence of its successive states (or positions): it also records all the different occurrences of the rules that were applied. In our framework, this is dealt with by describing the whole process as a single presheaf that describes a combinatorial object of higher dimension. Given a signature Σ that describes a set of objects (which are the presheaves over a category $\mathbb{C}$) and rewriting rules on these objects (given as a set $\mathcal{R}$ of spans of such objects), we build a pseudo double category $\mathbb{D}$ that models rewriting on this set of objects according to the given rewriting rules. This pseudo double category will have as objects the set of objects described by Σ, and as vertical morphisms from an object Y to an object X all the *execution traces* (also called *rewriting traces*) that correspond to rewriting X into Y according to the rules given by Σ. Processes are defined as particular cospans in $\widehat{\mathbb{C}[\mathcal{R}]}$ (the category of presheaves over some category $\mathbb{C}[\mathcal{R}]$ constructed from $\mathbb{C}$ and $\mathcal{R}$). Such a cospan $Y \to U \leftarrow X$ describes an execution trace whose starting position is X and whose final position is Y. We build $\mathbb{C}[\mathcal{R}]$ from $\mathbb{C}$ by adding an object r for each rewriting rule in $\mathcal{R}$ and some meaningful morphisms from objects of $\mathbb{C}$ to r. We then define *seeds*, which are the local shape of a rewriting step, by equipping each representable presheaf y_r with its initial and final positions X and Y (as given by the rewriting rule), thus forming a cospan $Y \to \mathsf{y}_r \leftarrow X$. Rewriting steps are then defined by embedding seeds into larger positions (thus allowing rewriting steps to occur in context). This whole process is reminiscent of Example 2.2.48. Finally, rewriting traces are defined to be compositions of rewriting steps in the bicategory of cospans in $\widehat{\mathbb{C}[\mathcal{R}]}$. We then derive the unfolding U_X of any object X as the colimit of all the rewriting traces whose starting position is X.

Under some assumptions on the category $\mathbb{C}$, the presheaves over $\mathbb{C}[\mathcal{R}]$ enjoy a graphical notation that is both close in spirit to string diagrams, and also inspired by that of graphs (but with higher-dimensional edges and multiple types of edges). For this reason, we usually call them *higher-dimensional* graphs or string diagrams (though none of these terms are actually very accurate).

In Section 3.2, we show how to derive a pseudo double category from a signature. We start by describing how we describe states (or positions) in Section 3.2.1, then the dynamics of the execution in Section 3.2.2, how to organise this into a pseudo double category $\mathbb{D}$ in Section 3.2.3, and how to derive a category of execution traces from $\mathbb{D}$ in Section 3.2.4. Finally, we show to describe the unfolding of an object in Section 3.3.

3.2 From Signatures to Pseudo Double Categories

This whole section is devoted to explaining the construction in detail. We also give applications and illustrate the construction on a running example.

3.2.1 Signatures and Positions

Required: 2.2.5.
Recommended: $\varnothing$.

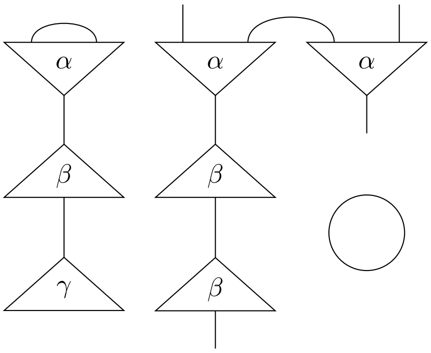
We start from basic data, which we call *signatures* and set out to create pseudo double categories of concurrent traces from them. We give three examples our techniques apply to, as well as a concrete running example on a particular Petri net.

Let us first give notations for our examples.

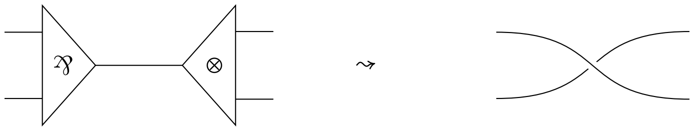
Notation 3.2.1. *A* Petri net *is a tuple* (P,T,δ,M)*, where* P *is a set of* places*,* T *is a set of* transitions*,* $\delta{:}T \to \mathbb{N}^P \times \mathbb{N}^P$ *is the* transition function*, and* $M{:}P \to \mathbb{N}$ *is the* initial marking*. Very briefly, for any marking* $M{:}P \to \mathbb{N}$*,* $M(p)$ *is the number of tokens in the place* p*, and for any transition* t*,* $\delta(t) \in \mathbb{N}^P \times \mathbb{N}^P$ *can be written as* $\delta(t) = (\delta_{cons}(t), \delta_{crea}(t))$*, where the first component associates to each place* p *the number of tokens of* p *consumed by firing the transition* t *and the second component associates to each place* p *the number of tokens created by firing* t*.*

A graph rewriting system $\mathcal{S}$ *is a set of spans* $r = L_r \leftarrow I_r \rightarrow R_r$ *of graphs describing the possibility to rewrite* L_r *into* R_r *(how this behaves on larger graphs that contain a copy of* L_r *depends on the exact formalism used).*

Finally, our third example is that of interaction systems *[65], which are akin to graph rewriting for* interaction nets*. Informally, an interaction net is some kind of circuit containing "gates" called* cells *whose types are given by* symbols*. Each symbol has an arity that determines the number of incoming* ports *for cells of this type. Cells always have a unique outgoing port. An interaction net is a circuit given by a number of cells and* free ports*, where ports are connected by* wires*, which are connected to exactly two or zero ports. For example, here is a drawing of an interaction net with symbols* α*,* β*, and* γ*, with arities* 2*,* 1*, and* 0 *respectively:*



It has seven cells, eleven wires (among which one is connected to no port and drawn as a loop) and four free ports. Formally, given a pair (A,a) *of a set* A *of symbols and an arity function* $a{:}A \to \mathbb{N}$*, an interaction net is a tuple* (C,l,W_0,W_2,F,p) *where* C *is the set of cells,* $l{:}C \to A$ *is the labelling of cells,* W_0 *is the set of wires connected to* 0 *ports,* W_2 *is the set of wires connected to* 2 *ports,* F *is the set of free ports, and* $p{:}W_2 \to \mathcal{P}_2(F + \sum_{c\in C} a(l(c)) + 1)$ *is the function that maps each wire to the ports it is connected to (this function must verify that all ports have a unique wire that is connected to it). Now, an interaction system* I *is a set of rules of the form* (x,y,σ)*, where* x *and* y *are symbols and* σ *is a partition of* $a(x) + a(y)$ *into pairs. Such a rule should be understood as: if the outgoing ports of two cells labelled* x *and* y *are linked by a wire, then the two cells may "disappear" together with the wire between their outgoing ports, and their incoming ports are linked according to* σ*. The most well-known rule of this form comes from interaction nets in linear logic, where* $⅋$ *and* $\otimes$ *cells reduce as below when faced with each other.*



Our construction starts from the basic notion of *signature*:

Definition 3.2.2 (Signature)**.** *A* signature Σ *is given by:*

- *a* base category $\mathbb{C}$ *that describes positions (more precisely, positions are presheaves over* $\mathbb{C}$*),*
- *a finite set* $\mathcal{R}$ *of spans* $Y \xleftarrow{w} I \xrightarrow{u} X$ *in the category* $\widehat{\mathbb{C}}$ *of presheaves over* $\mathbb{C}$*, called* rules*, which describe the possibility to rewrite* X *into* Y*,*
- *and a* starting position S *in* $\widehat{\mathbb{C}}$*.*

Example 3.2.3. *We here show how the three motivating examples can be described in our framework.*

- *In the case of Petri nets, the Petri net* (P, T, δ, M) *is modelled by:*
  - *the base category* $\mathbb{C} = P$*, where* P *is viewed as the discrete category with* $|P|$ *elements and only identity morphisms (note that presheaves on* $\mathbb{C}$ *indeed model markings of the Petri net),*
  - *if* $\delta = \langle \delta_{cons}, \delta_{crea} \rangle$*,* $\mathcal{R}$ *contains a rule* $Y_t \leftarrow \varnothing \rightarrow X_t$ *for each transition* $t \in T$*, where* Y_t *and* X_t *are resp. the multisets* $\delta_{crea}(t)$ *and* $\delta_{cons}(t)$*, seen as presheaves,*
  - *the starting position is* $S = M$*, seen as a presheaf.*
- *In the case of graph rewriting, rewriting a graph* G *according to a given set of rewriting rules* $\mathcal{S}$ *given as spans* $r = L_r \leftarrow I_r \rightarrow R_r$ *is modelled by:*
  - *the base category is the category with two objects* e, v *and two non-trivial arrows* $s, t \colon v \rightarrow e$ *(again, note that presheaves over* $\mathbb{C}$ *are indeed graphs),*
  - $\mathcal{R}$ *contains a rule* $R_r \leftarrow I_r \rightarrow L_r$ *for each* $r \in \mathcal{S}$*, where* I_r*,* L_r*, and* R_r *are seen as presheaves over* $\mathbb{C}$*,*
  - *the starting position is* $S = G$*, where* G *is seen as a presheaf over* $\mathbb{C}$*.*
- *Given a set* A *of symbols and an arity function* a*, rewriting an interaction net* (C, l, W_0, W_2, F, p) *according to an interaction system* I *given as a set of rules* (x, y, σ) *is modelled by:*
  - *a base category* $\mathbb{C}$ *that consists of one object* $*$ *for wires and an object* x *for each symbol* x *in* A*, as well as a morphism* $t \colon * \rightarrow x$ *and morphisms* $s_i \colon * \rightarrow x$ *for each* $i \in a(x)$*,*
  - $\mathcal{R}$ *contains a rule* $Y_r \xleftarrow{w_r} I_r \xrightarrow{u_r} X_r$ *for each rule* $r = (x, y, \sigma)$ *in* I*, where*
    - I_r *is coproduct* $(a(x) + a(y)) \cdot *$*,*[1]

[1] Here, $*$ is actually y_*, the representable presheaf on $*$. We will often implicitly identify an object of a category and its representable presheaf and let the context disambiguate.

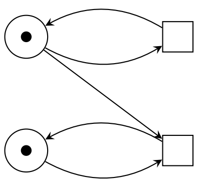


Figure 3.1: Our running example.

- *X_r is the pushout*

$$\begin{tikzcd}
2 \cdot * \arrow[r, "t+t"] \arrow[d, "\nabla"'] & x+y \arrow[d] \arrow[dl, phantom, "\urcorner", very near end] \\
* \arrow[r] & X_r
\end{tikzcd}$$

- *Y_r is the coproduct $((a(x)+a(y))/2)\cdot *$,*
- *$u_r\colon I_r \xrightarrow{[s_i]_{i\in a(x)}+[s_i]_{i\in a(y)}} x+y \to X_r$,*
- *$w_r\colon I_r \to Y_r$ is the quotient by σ (for all pairs (i,j) in σ, i and j have the same image),*

– *the starting position is the codomain of the coequaliser of*

$$W_0 \cdot * + W_2 \cdot * \rightrightarrows W_0 \cdot * + \textstyle\sum_{c\in C} c + F \cdot *$$

Remark. *The description of interaction systems as presheaves looks excruciatingly painful, but it is actually more due to the initial description of interaction nets (as tuples) than to the translation itself. Indeed, interaction nets are basically graphs, which are very easy to describe as presheaves, and the rules are similarly simple to describe.*

We could easily add some more features to these examples without increasing the complexity of the representation. For example, in the case of Petri nets, we could model "read arcs" by changing the middle component of the span (and therefore Y_t and X_t as well) to include the place the transition reads from. In the case of interaction systems, we could allow cells with several outgoing wires by changing the morphism t to n morphisms $t_1, \ldots, t_n$, or allow more complex shapes for the interaction rules.

Example 3.2.4. *We further specialise the example of Petri nets to the particular Petri net and marking shown in Figure 3.1. In our setting, this is modelled by:*

- *the set $\{a,b\}$ as its base category, where a represents the top-left place and b the bottom-left one,*
- *rules $a \leftarrow \varnothing \to a$ and $b \leftarrow \varnothing \to a+b$, which represent the top-right and bottom-right transitions respectively,*
- *the starting position is $a+b$.*

We will also use a more graphical notation for positions, i.e., presheaves over the set $\{a,b\}$. We could of course, as is usually done for Petri nets, represent a *position X by drawing a circle for each place and drawing as many bullets inside the circle that corresponds to a place p as there are elements in $X(p)$. However, this will prove quite impractical in our case, so we use a different representation. We will represent tokens in the place a by bullets and tokens in the place b by circles. For example,* $\bullet \quad \bullet \quad \circ$ *represents a marking with two tokens in a and one token in b.*

3.2.2 Execution Steps and Traces

Required: 3.2.1.
Recommended: 2.2.1,2.2.4.

Our base category $\mathbb{C}$ is only used to model positions. To represent dynamics in the same category, we augment $\mathbb{C}$ with objects that represent this dynamics. For each rule $r \in \mathfrak{R}$, we define a new category $\mathbb{C}_r$ that has the same objects as $\mathbb{C}$, plus an additional object r and morphisms from some objects of $\mathbb{C}$ to that new object.

Definition 3.2.5. *For any presheaf U, $\mathbb{C}[U]$ is the following cocomma category:*

$$\begin{tikzcd}
\mathrm{el}(U) \arrow[r, "!"] \arrow[d, "\pi_U"'] \arrow[dr, phantom, "\xRightarrow{\lambda}"] & 1 \arrow[d, "\ulcorner u \urcorner"] \\
\mathbb{C} \arrow[r, "i_U"'] & \mathbb{C}[U].
\end{tikzcd}$$

In more simple terms, $\mathbb{C}[U]$ has the same objects as $\mathbb{C}$, plus an additional object, say u. Between two objects of $\mathbb{C}$, $\mathbb{C}[U]$ has the same morphisms as $\mathbb{C}$, and the only morphism from u to itself is the identity morphism. For c an object of $\mathbb{C}$, there is a morphism $\bar{f}: c \to u$ for each morphism $f: c \to U$ (here again, we identify c and y_c), and there are no morphisms from u to c. Composition is defined in the obvious way.

The categories $\widehat{\mathbb{C}[U]}$ will describe some form of dynamics over positions in $\widehat{\mathbb{C}}$. We now want to describe presheaves over $\mathbb{C}[U]$ and their links with presheaves over $\mathbb{C}$. This will help us understand how to use them to describe dynamics.

For each presheaf X over $\mathbb{C}$, there is a presheaf $\overline{X} = \sum_{i_U}(X) = \mathrm{Lan}_{i_U^{op}}(X)$ over $\mathbb{C}[U]$. In simple terms, for each object c of $\mathbb{C}$, $\overline{X}(c)$ is isomorphic to $X(c)$, and $\overline{X}(u)$ is empty. Therefore, we will often denote X instead of $\overline{X}$ when it is obvious that the presheaf is over $\mathbb{C}[U]$.

Proposition 3.2.6. *Each presheaf $\overline{X}$ is the colimit of* $\mathrm{el}(X) \xrightarrow{\pi_X} \mathbb{C} \xrightarrow{i_U} \mathbb{C}[U] \xrightarrow{\mathsf{y}} \widehat{\mathbb{C}[U]}$.

Proof. We first show that $\sum_{i_U}(\mathsf{y}_c) \cong \mathsf{y}_{i_U(c)}$:

$$\textstyle\sum_{i_U}(\mathsf{y}_c)(d) \cong \int^{c'} \mathsf{y}_c(c') \times [d, i_U(c')] \cong \int^{c'} [c', c] \times \mathsf{y}_{i_U(c')}(d) \cong \mathsf{y}_{i_U(c)}(d)$$

by the coend formula for left Kan extension and the co-Yoneda lemma. Now we have:

$$\begin{aligned}\textstyle\sum_{i_U}(X) &\cong \textstyle\sum_{i_U}\left(\int^{c} X(c) \times y_c\right) \cong \int^{c} X(c) \times \sum_{i_U}(\mathsf{y}_c) \\ &\cong \int^{c} X(c) \times \mathsf{y}_{i_U(c)} \cong \mathrm{Lan}_{\mathsf{y}}(\mathsf{y} i_U)(X)\end{aligned}$$

by cocontinuity of left Kan extensions and the formula above.

Now, since

$$\begin{tikzcd}
\mathrm{el}(X) \arrow[r, "!"] \arrow[d, "\pi_X"'] \arrow[dr, phantom, "\Longrightarrow"] & 1 \arrow[d, "\ulcorner X \urcorner"] \\
\mathbb{C} \arrow[r, "\mathsf{y}"'] & \widehat{\mathbb{C}}
\end{tikzcd}$$

is a comma square and Kan extensions are pointwise in Cat, the composition of any left Kan extension $\mathrm{Lan}_{\mathsf{y}} F \colon \widehat{\mathbb{C}} \to \mathbb{C}'$ with $\ulcorner X \urcorner$ is again a left Kan extension. In particular, since $\mathrm{Lan}_{i_U^{op}}(-)$ is isomorphic to $\mathrm{Lan}_{\mathsf{y}}(\mathsf{y} i_U)$, which is a left Kan extension, it is also a left Kan extension, and therefore, so is $\mathrm{Lan}_{i_U^{op}}(-) \circ \ulcorner X \urcorner = \ulcorner \mathrm{Lan}_{i_U^{op}}(X) \urcorner = \ulcorner \overline{X} \urcorner$, so $\overline{X}$ is indeed the desired colimit. □

In particular, $\overline{U}$ is the colimit of $\mathrm{el}(U) \xrightarrow{\pi_U} \mathbb{C} \xrightarrow{i_U} \mathbb{C}[U] \xrightarrow{\mathsf{y}} \widehat{\mathbb{C}[U]}$. Moreover, we have the following natural transformation:

$$\begin{tikzcd}
\mathrm{el}(U) \arrow[r, "!"] \arrow[d, "\pi_U"'] \arrow[dr, phantom, "\overset{\lambda}{\Longrightarrow}"] & 1 \arrow[d, "\ulcorner u \urcorner"] \arrow[dr, "\ulcorner u \urcorner"] & \\
\mathbb{C} \arrow[r, "i_U"'] & \mathbb{C}[U] \arrow[r, "\mathsf{y}"'] & \widehat{\mathbb{C}[U]},
\end{tikzcd}$$

so, by universal property of the colimit, there is a unique morphism from $\overline{U}$ to u such that

$$\begin{tikzcd}
\mathrm{el}(U) \arrow[r, "!"] \arrow[d, "i_U \pi_U"'] \arrow[dr, phantom, "\Longrightarrow"] & 1 \arrow[r, equal] \arrow[d, "\ulcorner \overline{U} \urcorner"] \arrow[dr, phantom, "\Longrightarrow"] & 1 \arrow[d, "\ulcorner u \urcorner"] \\
\mathbb{C}[U] \arrow[r, "\mathsf{y}"'] & \widehat{\mathbb{C}[U]} \arrow[r, equal] & \widehat{\mathbb{C}[U]}
\end{tikzcd}$$

gives back the natural transformation above.

Definition 3.2.7. *We call $\partial u \colon U \to u$ be the morphism constructed above.*

In intuitive terms, U may be seen as a "hollow" version of u (or maybe conversely, u may be seen as a "full" version of U). They agree on every object in $\mathbb{C}$, but U is empty over the new object u, while u is a singleton over it. We may think of u as "giving a name" to the shape U by turning U into a single entity: the representable presheaf over u. ∂u is then just the morphism that adds the "filling" to U.

Definition 3.2.8. *For any rule $r \in \mathcal{R}$, let $\mathbb{C}_r$ be $\mathbb{C}[M_r]$, where M_r is defined as the following pushout:*

$$\begin{tikzcd}
I_r \arrow[r, "w_r"] \arrow[d, "u_r"'] & Y_r \arrow[d] \\
X_r \arrow[r] & M_r. \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

$\mathbb{C}_r$ is defined as any $\mathbb{C}[U]$, and, if we call r its new object, then the morphisms from c to r are as defined below:

- for each $f \colon \mathsf{y}_c \to X_r$ in $\widehat{\mathbb{C}}$, add a morphism $t_f \colon c \to r$ in $\mathbb{C}_r$,
- for each $f \colon \mathsf{y}_c \to Y_r$ in $\widehat{\mathbb{C}}$, add a morphism $s_f \colon c \to r$ in $\mathbb{C}_r$,
- for each $f \colon \mathsf{y}_c \to I_r$ in $\widehat{\mathbb{C}}$, add a morphism $\tilde{f} \colon c \to r$ in $\mathbb{C}_r$,
- quotient these morphisms by $\tilde{f} = t_{u_r f}$, $\tilde{f} = s_{w_r f}$, $t_{fg} = t_f g$, and $s_{fg} = s_f g$.

Remark. *In fact, we have only added morphisms t_f and s_f, since each morphism $\tilde{f}$ will be identified with $t_{u_r f}$ and $s_{w_r f}$.*

Building on the discussion above about "giving a name" and "filling" presheaves, if we see M_r as some sort of interaction or dynamics, then $\mathbb{C}_r$ is a category that can describe positions (because it contains $\mathbb{C}$), but can also speak about the dynamics described by M_r, because it contains a new object designed to fit this very purpose.

Definition 3.2.9. *In $\widehat{\mathbb{C}_r}$, we call $Y_r \xrightarrow{s_r} r \xleftarrow{t_r} X_r$ the cospan obtained by composing each leg of $Y_r \to M_r \leftarrow X_r$ with ∂r.*

Proposition 3.2.10. *$Y_r \xrightarrow{s_r} r \xleftarrow{t_r} X_r$ is such that*

$$\begin{tikzcd}
I_r \arrow[r, "w_r"] \arrow[d, "u_r"'] & Y_r \arrow[d, "s_r"] \\
X_r \arrow[r, "t_r"'] & r
\end{tikzcd} \tag{3.1}$$

commutes.

Proof. It is the composition of a commutative square with ∂r. □

We finally define $\mathbb{C}[\mathcal{R}]$, the base category our pseudo double category will be built on.

Definition 3.2.11. *For a signature $\Sigma = (\mathbb{C}, \mathcal{R}, S)$, we define the base category $\mathbb{C}[\mathcal{R}]$ as the wide pushout of $\mathbb{C} \xrightarrow{i_{M_r}} \mathbb{C}_r$ for all rules r in $\mathcal{R}$.*

In more simple terms, this is the category $\mathbb{C}$ with all the objects r in $\mathcal{R}$ and corresponding morphisms.

Example 3.2.12. *In our running example, if we call the first rule u and the second one v, we have that $\mathbb{C}[u, v]$, the base category of the corresponding pseudo double category will have:*

- *as objects: the set $\{a, b, u, v\}$,*
- *as non-trivial morphisms: $t_{id_a} \colon a \to u$, $s_{id_a} \colon a \to u$, $t_{i_l} \colon a \to v$, $t_{i_r} \colon b \to v$, $s_{id_b} \colon b \to v$,*

or, up to renaming, and in more graphical terms:

$$\begin{tikzcd}
u & v \\
a \arrow[u, bend left, "t_a^u"] \arrow[u, bend right, "s_a^u"'] \arrow[ur, "t_a^v"] & b. \arrow[u, bend left, "t_b^v"] \arrow[u, bend right, "s_b^u"']
\end{tikzcd}$$

3.2.3 Organising Traces into a Pseudo Double Category

Required: 3.2.2, 2.2.9.
Recommended: ∅.

We now build a pseudo double category that describes rewriting on the signature. More precisely, the objects of this pseudo double category will be positions, horizontal morphisms describe "spatial inclusion", i.e., how positions may be included in larger positions, and vertical morphisms are "rewriting traces", i.e., how a position can reach another position after a finite number of execution steps.

First of all, we need to define how we model an execution step.

In order to do this, we first define *seeds*, which are the basic shapes of execution steps.

Definition 3.2.13. *The* seed *that corresponds to the rule* $r = Y_r \xleftarrow{w_r} I_r \xrightarrow{u_r} X_r$ *is the commuting square* (3.1)*.*

Definition 3.2.14. *An* execution step*, or* rewriting step*, is any cospan* $Y \to M \leftarrow X$ *obtained by taking a pushout of a seed along some monomorphism* $I_r \to Z$ *(where* Z *is necessarily of the form* $\overline{Z'}$ *if we want* X *and* Y *to be positions):*

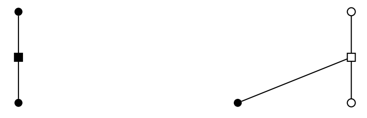
$$\begin{tikzcd}
 & Y_r \arrow[rrr] \arrow[dr] & & & Y \arrow[dr] \arrow[dl, phantom, "\ulcorner", very near start] & \\
 & & r \arrow[rrr] & & & M \arrow[l, phantom, "\ulcorner", very near start] \\
I_r \arrow[ur] \arrow[urr] \arrow[rrr] \arrow[dr] & & & Z \arrow[ur] \arrow[urr] \arrow[dr] & & \\
 & X_r \arrow[rrr] \arrow[uur] & & & X, \arrow[uur] \arrow[ul, phantom, "\llcorner", very near start] &
\end{tikzcd}$$

where $X \to M$ *and* $Y \to M$ *are obtained by universal property of pushout. We say that this rewriting step* rewrites X *into* Y*.*

Example 3.2.15. *In our running example, there are two seeds, which correspond respectively to* u *and* v*. They are the squares*

$$\begin{tikzcd}
\varnothing \arrow[r] \arrow[d] & a \arrow[d, "s_a^u"] \arrow[dl, phantom, "\ulcorner", very near end] & \text{and} & \varnothing \arrow[r] \arrow[d] & b \arrow[d, "s_b^v"] \arrow[dl, phantom, "\ulcorner", very near end] \\
a \arrow[r, "t_a^u"'] & u & & a + b \arrow[r, "{[t_a^v, t_b^v]}"'] & v.
\end{tikzcd}$$

We can represent them in a more graphical way. Recall the graphical notation for positions from Example 3.2.4. We can draw the seeds that correspond to u *and* v *as below.*



The initial position of each seed is drawn at the bottom, and its final position is drawn at the top, with another element in the middle that corresponds to the action of moving from the initial position to the final one itself.

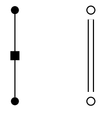
Remark. *These drawings actually correspond to drawing the category of elements of the seeds that correspond to u and v, just like drawing a graph corresponds to drawing the category of elements of its corresponding presheaf. Here, we have drawn the categories of elements of the presheaves u and v, with the convention that $X_u = a$ and $X_v = a + b$ are drawn at the bottom, while $Y_u = a$ and $Y_v = b$ are drawn at the top. Therefore, in all our drawings, "time flows upwards".*

Example 3.2.16. *Remember that our starting marking is $a + b$, so there are two possible rewriting steps on this marking, depicted below.*

$$\begin{tikzcd}
 & a \arrow[rrr] \arrow[dr] & & & a+b \arrow[dr] & \\
 & & u \arrow[rrr] & & & u+b \\
\varnothing \arrow[uur] \arrow[rrr] \arrow[dr] & & & b \arrow[uur] \arrow[dr] & & \\
 & a \arrow[uur] \arrow[rrr] & & & a+b \arrow[uur] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & b \arrow[rrr] \arrow[dr] & & & b \arrow[dr] & \\
 & & v \arrow[rrr] & & & v \\
\varnothing \arrow[uur] \arrow[rrr] \arrow[dr] & & & \varnothing \arrow[uur] \arrow[dr] & & \\
 & a+b \arrow[uur] \arrow[rrr] & & & a+b \arrow[uur] &
\end{tikzcd}$$

The first one corresponds to firing u on the marking $a + b$, which indeed gives back the marking $a + b$, while the second one corresponds to firing v on the same marking, which indeed gives the marking b.

We can draw these rewriting steps in a graphical way, very much like we did for seeds (and again, this corresponds to drawing the category of elements of the object we represent). Since the move that corresponds to v doesn't differ from the seed, it is represented exactly as in the example above, while the move that corresponds to u is drawn as below.



The top and bottom positions indeed correspond to the initial and final positions of the rewriting step, and the equal sign between the two tokens in b express that they are actually the same token, but that this token is present both in the initial and final positions (in the category of elements, there is of course only one object corresponding to the token in b, but we draw it twice with an equal sign between them for increased readability).

Definition 3.2.17. *An* execution trace *(or* rewriting trace*) is any cospan that is isomorphic to a finite composition of rewriting steps in the category of cospans on $\widehat{\mathbb{C}[\mathcal{R}]}$.*

Example 3.2.18. *For example, in our running example, the different rewriting traces are (isomorphic to) either:*

- *U_n for some n in $\mathbb{N}$, where U_n is the n-fold composition of the move that corresponds to u on the initial marking (note that they can indeed be composed because the final position is the same as the initial one),*
- *or V_n for some n in $\mathbb{N}$, where V_n is the composition of U_n with the move that corresponds to v.*

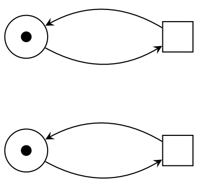
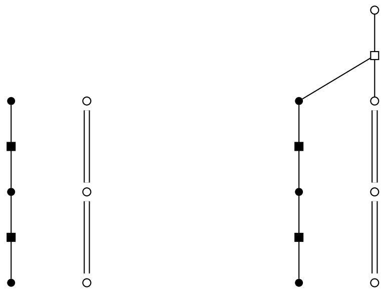


Figure 3.2: A less sequential Petri net.

They can be described in graphical terms, for example, the graphical descriptions of U_2 and V_2 are on the left and right below, respectively.



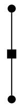
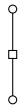
Example 3.2.19. *To see how the definition of execution trace is not sequential, but retains only the minimal causality information between rewriting steps, we have to introduce another Petri net, because the one from our running example is entirely sequential.*

Let us take as example the Petri net in Figure 3.2, which is much more parallel: indeed, the two transitions do not interact anymore. In this Petri net, the v transition (if we keep the same name as for the previous Petri net) is not $a+b \leftarrow \varnothing \rightarrow b$ anymore, but $b \leftarrow \varnothing \rightarrow b$. If we still call a, b, u, and v the different places and transitions, we have that the possible moves from the initial marking are isomorphic to one of the following two moves:

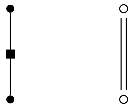
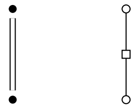
$$\begin{tikzcd}
 & a \arrow[rrr] \arrow[dr] & & & a+b \arrow[dr] & \\
 & & u \arrow[rrr] & & & u+b \\
\varnothing \arrow[uur] \arrow[rrr] \arrow[dr] & & & b \arrow[uur] \arrow[dr] & & \\
 & a \arrow[rrr] \arrow[uur] & & & a+b \arrow[uur] &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & b \arrow[rrr] \arrow[dr] & & & a+b \arrow[dr] & \\
 & & v \arrow[rrr] & & & a+v \\
\varnothing \arrow[uur] \arrow[rrr] \arrow[dr] & & & a \arrow[uur] \arrow[dr] & & \\
 & b \arrow[rrr] \arrow[uur] & & & a+b. \arrow[uur] &
\end{tikzcd}$$

A possible execution in the Petri net consists in first firing u, then firing v. It is the composition of the two cospans $a+b \rightarrow u+b \leftarrow a+b$ and $a+b \rightarrow a+v \leftarrow a+b$, which gives $a+b \rightarrow u+v \leftarrow a+b$, which is the same as the rewriting trace obtained by first firing v, and then firing u, which corresponds to the fact that there is no causality relation between firing u and firing v.

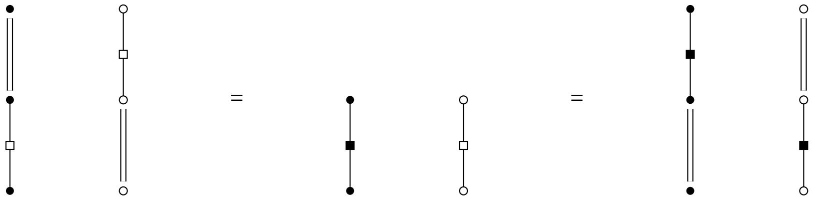
This can also be seen in graphical terms: if the seeds are drawn as

 *and*  ,

then the moves on the initial marking can be drawn as

 *and*  .

Therefore, the execution that fires u then v and the one that fires v then u can be drawn as on the left and right respectively, which can also be drawn as in the middle (these drawings are again the graphical representation of the categories of elements associated to the different presheaves):


.

We can now define the pseudo double category $\mathbb{D}$ that will describe processes:

- its objects are the positions, i.e., the presheaves over $\mathbb{C}$ (more precisely, they are the presheaves over $\mathbb{C}[\mathcal{R}]$ that are isomorphic to $\overline{X}$ for some presheaf X over $\mathbb{C}$),
- its horizontal morphisms are monomorphisms of positions,
- its vertical morphisms are rewriting traces,
- its double cells are monomorphisms of traces, i.e., commuting diagrams of the form

$$\begin{tikzcd}
Y \arrow[r, "l"] \arrow[d] & Y' \arrow[d] \\
U \arrow[r, "k"] & U' \\
X \arrow[r, "h"'] \arrow[u] & X', \arrow[u]
\end{tikzcd}$$

where h, k, and l are mono.

Remark. *We can draw a parallel between rewriting and games (and indeed, the ideas used here to model rewriting stem from* playgrounds*, which are at the base of the work described in this dissertation):*

- *the objects our rewriting system works on (in the case of Petri nets, markings) are akin to* positions,
- *rewriting steps (in the case of Petri nets, the firing of a transition) to* moves,
- *and rewriting traces (in the case of Petri nets, executions) to* plays.

3.2.4 The Category of Execution Traces

Required: 3.2.3,
Recommended: ∅.

In this section, we derive a category $\mathbb{E}(X)$ of execution traces that start from a given position X from our pseudo double category $\mathbb{D}$.

Definition 3.2.20. *The category $\mathbb{E}(X)$ is the category of rewriting traces over X:*

- *its objects are the lower legs of rewriting traces: for each rewriting trace $Y \xrightarrow{s} U \xleftarrow{t} X$, $X \xrightarrow{t} U$ is an object of $\mathbb{E}(X)$,*
- *the morphisms from $X \xrightarrow{t} U$ to $X \xrightarrow{t'} U'$ are monomorphisms $U \xrightarrow{f} U'$ such that $ft = t'$.*

The morphisms in $\mathbb{E}(X)$ correspond to "prefix" inclusion of rewriting traces, i.e., there is a morphism $U \to U'$ in $\mathbb{E}(X)$ if the rewriting trace U can be followed by another rewriting trace such that the whole rewriting trace is isomorphic to U'.

Remark. *We can make the statement above much more obvious by defining $\mathbb{E}(X)$ in terms of the pseudo double category $\mathbb{D}$ we just built above. The construction is as follows:*

- *the objects of $\mathbb{E}(X)$ are plays $Y \to U \leftarrow X$,*
- *a morphism from $Y \to U \leftarrow X$ to $Y' \to U' \leftarrow X$ is a pair of a morphism $Z \to V \leftarrow Y$ and a double cell*

$$\begin{tikzcd}
Z \arrow[r] \arrow[d, "V"'] & Y' \arrow[dd, "U'"] \\
Y \arrow[d, "U"'] \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & \\
X \arrow[r, equal] & X,
\end{tikzcd}$$

where such morphisms are quotiented by the equivalence relation generated by $(V, \alpha) \sim (V', \alpha')$ when there is a morphism $Z \to Z'$ and a double cell

$$\begin{tikzcd}
Z \arrow[r] \arrow[d, "V"'] \arrow[dr, phantom, "\overset{\gamma}{\Longrightarrow}"] & Z' \arrow[d, "V'"] \\
Y \arrow[r, equal] & Y
\end{tikzcd}$$

such that $\alpha = \alpha' \circ (U \bullet \gamma)$.

With this definition of $\mathbb{E}(X)$, it is much more obvious that morphisms of $\mathbb{E}(X)$ correspond to "prefix" (or "temporal") inclusions. However, we should now show that morphisms compose and that both definitions coincide. We do exactly this in Section 5.2.3, albeit on a different base category, but the techniques we use there also work in the case of Petri nets and our other examples.

Example 3.2.21. *In our running example, $\mathbb{E}(X)$ is composed of all the plays isomorphic to either U_n or V_n (as defined in Example 3.2.18), plus the morphisms that correspond to prefix inclusion, which are generated by:*

- *for each n, the only morphism $U_n \to U_{n+1}$ that fixes X,*
- *for each n, the only morphism $U_n \to V_n$ that fixes X.*

3.3 Unfolding

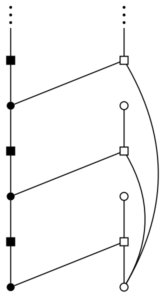
Required: 3.2.4,
Recommended: $\varnothing$.

Even though we will not use this kind of technique in the rest of this dissertation, it may be interesting to know that building the unfolding of an object using our techniques is extremely simple.

Definition 3.3.1. *The unfolding U_X of X is the colimit of the functor $\mathbb{E}(X) \hookrightarrow X/\widehat{\mathbb{C}[\mathcal{R}]} \xrightarrow{6} \widehat{\mathbb{C}[\mathcal{R}]}$.*

Remark. *Since $X \xrightarrow{id_X} X$ is an object of $\mathbb{E}(X)$, the unfolding U_X of X comes with a map $X \to U_X$ obtained by universal property of U_X.*

Example 3.3.2. *In our running example, if we draw the unfolding U_X of X, we obtain:*



,

and the morphism $X \to U_X$ maps X to the bottom of this drawing, which is exactly what the unfolding of our Petri net is supposed to look like.

3.4 Perspectives

The purpose of this chapter was to introduce the reader to the use of presheaves as combinatorial objects to represent execution traces in a concurrent way. We have shown through a running example that using this technique gives a faithful representation of executions in Petri nets. We have also shown that this representation is indeed concurrent, in the sense that, if two independent moves are played in a row, then the interpretation does not depend on whether the first one is played first or second.

The example of Petri nets is only used to give some intuition about the tools we use, and not meant to be an interesting example. Indeed, we have promised

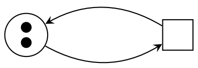
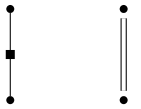


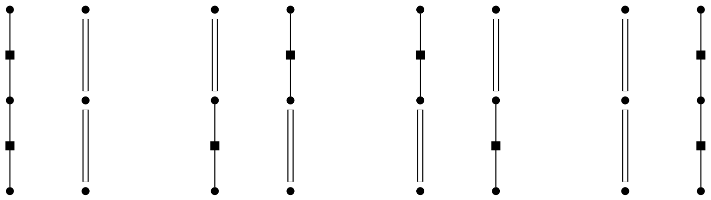
Figure 3.3: An unsafe Petri net.

that our representation of execution traces faithfully represents executions in Petri nets, and it is indeed the case in the examples we have given, but not in general. It actually only works for *safe* Petri nets (i.e., those Petri nets such that there is never more than one token in a given place). As a counter example, consider the Petri net in Figure 3.3. Since there is a single transition, there is at most a single execution trace of any given length on this Petri net. Now, let us consider our representation of execution traces for this Petri net. The base category $\mathbb{C}$ has two objects: one object a that represents the place and one object u that represents the transition; and two non-trivial morphisms: $t, s: a \to u$. The cospan that represents the transition is $a \xrightarrow{s} u \xleftarrow{t} a$, and the initial position of the Petri net is $2 \cdot a$. There is only one possible move on the initial position (up to isomorphism): $2 \cdot a \xrightarrow{s+a} u + a \xleftarrow{t+a} 2 \cdot a$. We may represent this transition graphically as:

 .

So far, so good.

A first problem becomes apparent when we look at plays of length greater than 1. Indeed, in such a play, each move can be played either on the left token or on the right token. A good thing is that this does not lead to an exponential blow-up, but to a linear one: there are $\lceil \frac{n+1}{2} \rceil$ plays of length $n > 0$ (again, up to isomorphism). For example, let us enumerate all the plays of length 2 up to isomorphism. There are four ways to compose this transition with itself, as depicted below.



But the two ways to compose drawn in the middle are isomorphic, as well as the two on the sides, which gives two non-isomorphic plays. In general, composing this transition n times with itself corresponds to choosing how many times the transition is applied to the left token, and how many times it is applied to the right one, so there are $\lceil \frac{n+1}{2} \rceil$ plays up to isomorphism.

Another problem arises when we look at $\mathbb{E}(2 \cdot a)$. There, there are even two non-isomorphic plays of length 1: the plays $u + a$ and $a + u$ are not isomorphic in $\mathbb{E}(2 \cdot a)$! Indeed, remember that a morphism between these two plays in $\mathbb{E}(2 \cdot a)$ is a pair (U, α) such that

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "U\bullet"'] & 2\cdot a \arrow[dd, "\bullet u+a"] \\
2\cdot a \arrow[r, Rightarrow, "\alpha"] \arrow[d, "a+u\bullet"'] & {} \\
2\cdot a \arrow[r, equal] & 2\cdot a
\end{tikzcd}$$

commutes. By unrolling the definitions, such a morphism is the pair of a cospan $X \xrightarrow{s_U} U \xleftarrow{t_U} 2{\cdot}a$ and a pair of monomorphisms $l\colon X \to 2{\cdot}a$ and $k\colon (u{+}a){\bullet}U \to a{+}u$ such that

$$\begin{tikzcd}
X \arrow[rrr, "l"] \arrow[dr, "s_U"'] & & & 2\cdot a \arrow[dd, "a+s"] \\
& U \arrow[dr] & & \\
2\cdot a \arrow[ur, "t_U"] \arrow[dr, "s+a"'] & & (u+a)\bullet U \arrow[r, "k"] & a+u \\
& u+a \arrow[ur] & & \\
2\cdot a \arrow[ur, "t+a"] \arrow[rrr, equal] & & & 2\cdot a \arrow[uu, "a+t"']
\end{tikzcd}$$

commutes, which is impossible: if we start from the bottom-left $2\cdot a$, through the top-left path, the "left-hand" a is sent to u, which must be sent to u (it cannot be sent to a because there are no morphisms $u \to a$ in $\mathbb{C}$), but through the bottom-right path, it is sent again to the "left-hand" a through the identity, and then to a, so this square cannot commute.

Both problems come from the fact that tokens are "named" in our presheaves: if X is a presheaf, then $X(a)$ is a set, and we can distinguish between the elements of this set. Firing a transition that consumes a token x is thus different from firing one that consumes another token x'. This means that there may be several representations of the firing of the same transition on the same marking in a Petri net. We know one possible way to solve this problem, but the answer is not elegant, and the construction of the base category becomes much more involved and less intuitive, so we only wrote the simple construction here, while perfectly aware of its limitations.

Notice however that this naming problem is limited to Petri nets, and it is not a problem at all in our other examples: in graph rewriting, the names are given to vertices and edges, and in interaction systems, the names are given to wires and cells, and, in both cases, it is important to distinguish between different objects of these classes, while no distinctions should be made between different tokens in the same place in a Petri net.

Chapter 4

Pseudo Double Categories and Concurrent Game Models

4.1 Motivation

In this chapter, we build an abstract framework to create sheaf models of programming languages based on string diagrams. The goal of this construction is to unify the previous constructions of game semantical sheaf models for CCS [50] and the π-calculus [29] and make the method used there general enough to create a variant of HON games (which is slightly harder than for CCS and the π-calculus for technical reasons), which we do in Chapter 5. Another hope is that this construction is general enough to encompass a large class of languages, thus facilitating the creation of new sheaf models.

This framework takes as input a *signature* describing the operational semantics of a language and outputs a *fibred pseudo double category* (i.e., a pseudo double category that possesses an additional property we call *fibredness*) that describes the language in terms of games based on string diagrams. These pseudo double categories are candidate models of the programming language we start from. The models built from these pseudo double categories may be seen as game semantical models because terms are interpreted as innocent strategies in some kind of game, or sheaf models because innocent strategies are sheaves for a Grothendieck topology induced by embedding views into plays.

The fibred pseudo double categories that are built this way enjoy a number of fundamental properties that make them a good framework to build game semantical models. They are a simplification of the notion of *playground* (but they do not enjoy all the good properties of playgrounds), which was introduced by Hirschowitz [50, 29] as a framework to study game semantics for concurrent languages. In particular, the property of fibredness is crucial to define composition in the category $\mathbb{E}(X)$ of plays whose starting position is X, which is necessary to define strategies.

Let us be a bit more precise. In our approach, the pseudo double category $\mathbb{D}$ we build from a signature represents the game as a whole. Its objects model positions in the game. For each pair (X, Y) of positions, its set of vertical morphisms $\mathbb{D}_v(Y, X)$ models all *plays* with initial position X and final position

Y. For all pairs of positions (X, X'), its set of horizontal morphisms $\mathbb{D}_h(X', X)$ roughly models all ways of embedding the position X' into X. Finally, given plays and morphisms as in

$$\begin{tikzcd}
Y' \arrow[r, "h"] \arrow[d, "u"', "\bullet" marking] & Y \arrow[d, "v", "\bullet" marking] \\
X' \arrow[r, "k"'] & X,
\end{tikzcd}$$

the set of cells $\mathbb{D}(h, u, k, v)$ models embeddings from u into v that preserve both the initial and final positions. E.g., u could describe the part of the play v which concerns players in X'. We can build the categories $\mathbb{E}(X)$ from $\mathbb{D}$ in a uniform way, no matter the shape of X.

Beyond providing a uniform construction for the categories $\mathbb{E}(X)$, this rich structure yields links between them, which greatly facilitate the development. Indeed, all plays $u{:}\,Y \mathrel{-\!\!\!\bullet\!\!\!\to} X$ induce a functor $u^*{:}\,\mathrm{Sh}(\mathbb{E}(X)) \to \mathrm{Sh}(\mathbb{E}(Y))$ between the corresponding categories of sheaves (which represent innocent strategies). Similarly, each horizontal morphism $h{:}\,X' \to X$ induces a restriction functor $h^*{:}\,\mathrm{Sh}(\mathbb{E}(X)) \to \mathrm{Sh}(\mathbb{E}(X'))$. These functors are semantically relevant: for any innocent strategy $S \in \mathrm{Sh}(\mathbb{E}(X))$, $u^*(S)$ denotes the "residual" of S after u, i.e., a description of how the players of X would behave according to S after playing u. A meaningful transition system on innocent strategies is then (roughly) given by triples $S \xrightarrow{M} M^*(S)$, for any move M in the game. The obtained transition system is shown in both of our previous models to be closely related to the operational semantics of the considered calculus. On the other hand, $h^*(S)$ denotes a *restriction* of S to X', i.e., a description of S for players in the subposition X' of X. This is useful for composing strategies: if a given position X is divided into two subpositions X' and X'', thought of as two teams, a composite (in a sense analogous to parallel composition in game semantics) of $S' \in \mathrm{Sh}(\mathbb{E}(X'))$ and $S'' \in \mathrm{Sh}(\mathbb{E}(X''))$ is any $S \in \mathrm{Sh}(\mathbb{E}(X))$ whose respective restrictions to X' and X'' give S' and S''.

Our first main contribution is a way to produce such fibred pseudo double categories automatically from more basic data. The kind of basic data we will use is the notion of *signature* defined in Section 4.3. From any *signature* S, we construct a pseudo double category $\mathbb{D}(\mathsf{S})$. Up to the verification of a few additional conditions on $\mathbb{D}(\mathsf{S})$, new sheaf models may thus be produced directly from nice signatures S.

As mentioned above, one crucial such condition, on which the very construction of our categories $\mathbb{E}(X)$ is based, is fibredness. Our second main contribution is then in Section 4.4 to prove that, under suitable hypotheses, $\mathbb{D}(\mathsf{S})$ satisfies this property. More precisely, we provide two results:

- Under a necessary and sufficient, but hard-to-verify condition essentially saying that fibredness is satisfied for all "generators" (called *seeds*) in S, we prove that $\mathbb{D}(\mathsf{S})$ is fibred (Theorem 4.4.30).

- We then exhibit sufficient, easier-to-verify conditions for the hard-to-verify condition above to hold (Theorem 4.4.39).

By plugging both results together, we obtain (Corollary 4.4.40) that any S satisfying the given sufficient conditions yields a fibred $\mathbb{D}(\mathsf{S})$.

4.2 Preliminaries

Required: 2.2.5.
Recommended: ∅.

In this preliminary section, we collect a few basic results about pushouts, pullbacks and monos in presheaf categories, which we will consider to be second nature in the sequel. The well-known pullback and pushout lemmas are not recalled, though widely used throughout. Similarly, let us merely recall that epis are stable under pullback, and that monos are stable under pushout in presheaf categories.

Let us start with an easy result about pullbacks along monos:

Lemma 4.2.1. *Any commuting square as below with j iso and m monic is a pullback:*

$$\begin{tikzcd}
A \arrow[r] \arrow[d, "j"'] & B \arrow[d, hook, "m"] \\
C \arrow[r] & D.
\end{tikzcd}$$

Proof. A simple diagram chase. □

Our second result is specific to sets:

Lemma 4.2.2. *Any commuting square of the form below left is a pullback if each rectangle as below right is:*

$$\begin{tikzcd}
\sum_{i\in I} A_i \arrow[r, "{[f_i]_{i\in I}}"] \arrow[d, "\sum_{i\in I} k_i"'] & A \arrow[d, "k"] \\
\sum_{i\in I} B_i \arrow[r, "{[g_i]_{i\in I}}"'] & B
\end{tikzcd}
\qquad
\begin{tikzcd}
A_i \arrow[r, "f_i"] \arrow[d, "k_i"'] & A \arrow[d, "k"] \\
B_i \arrow[r, "g_i"'] & B.
\end{tikzcd}$$

Proof. Straightforward. □

Our third result is an instance of the other pullback lemma [91].

Lemma 4.2.3 (Another pullback lemma)**.** *In any presheaf category, for any commuting diagram as below with e epi, if the outer rectangle and the left-hand square are pullbacks, then so is the right-hand square:*

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & B \arrow[r] \arrow[d] & C \arrow[d] \\
X \arrow[r, "e"'] & Y \arrow[r] & Z.
\end{tikzcd}$$

Proof. An immediate consequence of [91, Theorem 1], given that, in any presheaf category, epimorphisms are stable under pullbacks and (Epi, Mono) is a factorisation system, so all epimorphisms are strong. □

Let us continue by recalling the adhesivity properties of presheaf categories [64].

Lemma 4.2.4. *In any presheaf category, any pushout along a mono is also a pullback. Explicitly, any pushout square*

$$\begin{tikzcd}
A \arrow[r, hook, "m"] \arrow[d] & B \arrow[d] \\
C \arrow[r] & D \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

with m mono is also a pullback.

Lemma 4.2.5 (Adhesivity)**.** *In any presheaf category, for any commuting cube*

$$\begin{tikzcd}
I \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, very near start, "\lrcorner"] \arrow[ddr, phantom, very near start, "\lrcorner"] & & B \arrow[dr] \arrow[dd] & \\
& A \arrow[rr] \arrow[dd] & & C \arrow[dd] \\
I' \arrow[rr, hook] \arrow[dr] & & B' \arrow[dr] & \\
& A' \arrow[rr] & & C' \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

with the marked pullbacks, mono and pushout, all vertical faces are pullbacks if and only if the top face is a pushout.

Proof. By [64, Example 6 and Proposition 8 (iii)]. □

Let us finish with a similar-looking statement, which has in fact more to do with extensivity [19] of Set than with adhesivity.

Lemma 4.2.6. *In* Set*, for any commuting cube*

$$\begin{tikzcd}
I \arrow[rr] \arrow[dr] \arrow[dd] \arrow[ddr, phantom, very near start, "\lrcorner"] & & B \arrow[dr] \arrow[dd] & \\
& A \arrow[rr] \arrow[dd] & & C \arrow[dd, "f"] \arrow[ul, phantom, very near start, "\ulcorner"] \\
I' \arrow[rr] \arrow[dr] & & B' \arrow[dr] & \\
& A' \arrow[rr] & & C' \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

with the marked pushouts and pullback,

- *if $I' \to B'$ is injective then the front square is a pullback, and*
- *if all arrows except perhaps f are injective, then f also is.*

Proof. The following proof is due to Paweł Sobociński (private communication). In Set, the map $m: I' \to B'$, being injective, may be written as a coproduct injection $m: I' \to I' + X'$. But, injective maps being stable under pullback and coproduct injections being stable under pushout, the whole cube may be written as

$$\begin{tikzcd}
I \arrow[rr] \arrow[dr] \arrow[dd, "g"'] \arrow[dr, phantom, "\lrcorner", very near start] & & I + X \arrow[dr] \arrow[dd, "g+k" near end] & \\
& A \arrow[rr] \arrow[dd, "h" near start] & & A + X \arrow[dd, "f"] \arrow[ul, phantom, "\ulcorner", very near start] \\
I' \arrow[rr] \arrow[dr] & & I' + X' \arrow[dr] & \\
& A' \arrow[rr] & & A' + X' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

with $f = h + k$. This in particular shows that injectivity of h and k entails injectivity of f. Let us now show that the front face is a pullback. Indeed, it is the pasting of both left-hand squares below:

$$\begin{tikzcd}
A \arrow[r] \arrow[d, equal] & A + X \arrow[d, "A+k"] & X \arrow[l] \arrow[d, "k"] \\
A \arrow[r] \arrow[d, "h"'] & A + X' \arrow[d, "h+X'"] & X' \arrow[l] \arrow[d, equal] \\
A' \arrow[r] & A' + X' & X'. \arrow[l]
\end{tikzcd}$$

All rows being coproduct injections, by extensivity all squares are pullbacks, hence so is the face of interest by the pullback lemma. □

Corollary 4.2.7. *For any commuting cube as in Lemma 4.2.6 in any presheaf category with the marked pushouts and pullback,*

- *if $I' \to B'$ is monic then the front square is a pullback, and*
- *if all arrows except perhaps f are monic, then f also is.*

Proof. Monos and pullbacks are pointwise in presheaf categories. □

4.3 Signatures for Pseudo Double Categories

In this section, we define the notion of signature and illustrate it with the example of the π-calculus. We start by generalising the method used in the previous sheaf models to define the notion of signature. We then abstractly give the construction of the pseudo double category $\mathbb{D}(\mathsf{S})$ associated to any signature S. Finally, we motivate fibredness by showing how it occurs in the definition of relevant categories of plays.

Any signature S will comprise a base category $\mathbb{C}$, and $\mathbb{D}(\mathsf{S})$ will be a sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$ (as defined in Example 2.2.93). Very roughly, $\mathbb{C}$ will be equipped with a notion of dimension, and S will consist of a selection of cospans

$$Y \xrightarrow{s} M \xleftarrow{t} X \tag{4.1}$$

in $\widehat{\mathbb{C}}$ viewed as morphisms $Y \mathrel{-\!\!\bullet\!\!\to} X$ in $\mathsf{Cospan}(\widehat{\mathbb{C}})_v$, where X and Y have dimension at most 1 and M may have arbitrary dimension. We will call these cospans *seeds*. The intuition is that presheaves of dimension ≤ 1 model *positions* in a game (they are essentially graphs), while higher-dimensional presheaves model the dynamics of the game. Thus each cospan (4.1) models a *play*, starting in position X and ending in position Y, and M models *how* the play evolves from

X to Y. Up to some technicalities, $\mathbb{D}(\mathsf{S})$ is the smallest sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$ whose objects are positions and whose vertical morphisms contain seeds.

4.3.1 A Signature for the π-Calculus

Required: 2.2.5.
Recommended: 3.2.2.

In this section, we define a signature for the π-calculus to explain and motivate the abstract notion of signature we will use in the rest of the chapter.

Method

The method used in previous work to design games proceeds in four stages:

(i) Design a base category $\mathbb{C}_1$ over which finitely presentable presheaves will model positions in the game.

(ii) Select a collection of spans of monomorphisms modelling "typical" moves in the game. In each selected span $Y \xleftarrow{w} Z \xrightarrow{u} X$,

- X denotes the initial position of the move,
- Y denotes the final position of the move,
- Z denotes the part of the position which remains unchanged during the move.

This is in line with the double pushout approach to graph rewriting [33].

(iii) The next step is characteristic of our approach, and allows us to give a causal representation of plays not as spans but rather as cospans (on a richer base category). The obtained representation is significantly simpler than analogous, span-based representations of rewrite sequences [10], and a simple example is treated in detail in Chapter 3. It proceeds by augmenting $\mathbb{C}_1$ to a category $\mathbb{C}$ over which finitely presentable presheaves will also model plays. This goes by adding one new object μ_S for each selected span $S = (Y \xleftarrow{w} Z \xrightarrow{u} X)$, in a way that

$$\begin{tikzcd}
Z \arrow[r, hook, "w"] \arrow[d, hook, "u"'] \arrow[dr, phantom, very near start, "\lrcorner"] & Y \arrow[d, hook, "s"] \\
X \arrow[r, hook, "t"'] & \mu_S
\end{tikzcd}$$

is a pullback (with the marked monos). The resulting cospans $Y \xrightarrow{s} \mu_S \xleftarrow{t} X$ are called *seeds* of the game. In fact, this description of seeds is slightly naive, as it implies that any two seeds are independent from each other, a simplification that we cannot afford if we aim at fibredness. So we should also describe morphisms between the new objects, and select our spans in a compatible way, which essentially means that they should induce a functor $\mathsf{S}\colon \mathbb{C}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{C}})_H$, where $\mathbb{C}_{|\geq 2}$ is the full subcategory of $\mathbb{C}$ spanning the new objects.

(iv) In the last stage, roughly (see Section 4.3.3 for details), we construct the smallest sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$

- whose objects are positions,
- whose vertical morphisms include all identities and seeds, and
- which contains all pushouts of the form

$$\begin{tikzcd}
id^{\bullet}_{Z} \arrow[r, "id^{\bullet}_{h}"] \arrow[d] & id^{\bullet}_{Z'} \arrow[d] \\
\mathsf{S}(\mu_S) \arrow[r] & M \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

  for all selected spans $S = (Y \leftarrow Z \rightarrow X)$ and morphisms $h{:}\, Z \rightarrow Z'$ with Z' a position.

The last point intuitively says that moves (modelled by seeds) may occur in context.

We will handle the first three stages here to produce our example signature S_π. The signature we get is exactly the one we used for our previous model of the π-calculus [29]. By applying the last step to S_π, we get exactly the pseudo double category that our model of the π-calculus is built on. Another example is given in Section 5.2, where we define a signature for HON games.

Operational description of the language and informal description of the game

We start by giving an operational description of the π-calculus that will guide us through steps *(i)–(iii)*.

The π-calculus we study is slightly unusual in the sense that:

- its terms are possibly infinite, which spares us the need to define constructors for replication or to put recursion in our language,
- the reduction of terms is done in terms of a *chemical abstract machine*, in the style of Berry and Boudol [12].

Definition 4.3.1. Processes *of the π-calculus are terms coinductively generated by the grammar*

$$\frac{\gamma \vdash_{\mathsf{g}} P_1 \qquad \ldots \qquad \gamma \vdash_{\mathsf{g}} P_n}{\gamma \vdash \sum_{i \in n} P_i} \qquad\qquad \frac{\gamma \vdash P \qquad \gamma \vdash Q}{\gamma \vdash P \,|\, Q}$$

$$\frac{\gamma, a \vdash P}{\gamma \vdash_{\mathsf{g}} \nu a.P} \qquad \frac{\gamma \vdash P}{\gamma \vdash_{\mathsf{g}} \tau.P} \qquad \frac{a \in \gamma \qquad \gamma, b \vdash P}{\gamma \vdash_{\mathsf{g}} a(b).P} \qquad \frac{a, b \in \gamma \qquad \gamma \vdash P}{\gamma \vdash_{\mathsf{g}} \bar{a}\langle b \rangle.P} \;,$$

where

- *we have two judgements,* $\vdash$ *for* processes *and* $\vdash_{\mathsf{g}}$ *for* guarded processes*;*
- γ *ranges over finite sets of natural numbers, and*

- *γ, a is defined if and only if $a \notin \gamma$ and then denotes $\gamma \uplus \{a\}$.*

We denote by Pi_γ the set of all processes of the form $\gamma \vdash P$.

To define the reduction of our π-calculus, we first need to define the *configurations* of the chemical abstract machine:

Definition 4.3.2. *A* configuration *is a pair of a finite set of natural numbers γ and a finite multiset of processes $\gamma \vdash P_i$. We write such a configuration $\langle \gamma \| P_1, \ldots, P_n \rangle$.*

The notation below is a formal definition to make sense of the intuitive notation $P + Q$ when P is a guarded process and Q is a sum, and which does exactly what one would expect:

Notation 4.3.3. *For any $\gamma \vdash_{g} P$, $\gamma \vdash Q$ of the form $\sum_{i \in n} Q_i$, and injection $h \colon n \hookrightarrow n+1$, we denote by $P +_h Q$ the sum $\sum_{j \in n+1} P_j$, where $P_{h(i)} = Q_i$ for all $i \in n$ and $P_k = P$, for k the unique element of $(n+1) \smallsetminus \mathrm{Im}(h)$.*

We finally define the reduction of configurations:

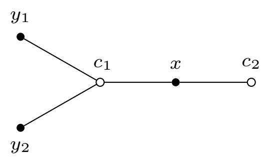
$$\frac{}{\langle \gamma \| a(b).P +_h R, \bar{a}\langle c \rangle.Q +_{h'} R' \rangle \to \langle \gamma \| P[b \mapsto c], Q \rangle} \qquad \frac{}{\langle \gamma \| P \mid Q \rangle \to \langle \gamma \| P, Q \rangle}$$

$$\frac{}{\langle \gamma \| \tau.P +_h R \rangle \to \langle \gamma \| P \rangle} \qquad \frac{}{\langle \gamma \| \nu a.P +_h R \rangle \to \langle \gamma, a \| P \rangle}$$

$$\frac{\langle \gamma_1 \| S_1 \rangle \to \langle \gamma_2 \| S_2 \rangle}{\langle \gamma_1 \| S \cup S_1 \rangle \to \langle \gamma_2 \| S \cup S_2 \rangle}$$

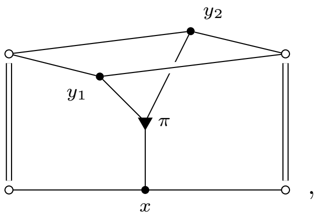
Let us now informally describe the game based on this operational description of our π-calculus. A *position* will be a finite hypergraph (i.e., a graph in which edges do not link two vertices but an arbitrary number of them). In a position, we call nodes *players* and edges *channels.* A player who is linked to n channels is a placeholder for a process $\gamma \vdash P$ with n free communication channels, i.e., where $|\gamma| = n$. A channel that is linked to n players is thought of as a communication channel that is shared by n processes. Our game is intrinsically multi-party in the sense that there are several players in a play.

Example 4.3.4. *The position below is a position with three players x, y_1, and y_2, and two channels c_1 and c_2. The first channel c_1 is shared between all three players, while the second channel c_2 is private to x.*



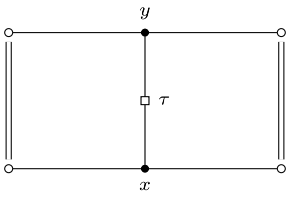
Let us now describe moves in the game. They are typically given by the operational description of the language, which, in the case of the π-calculus, are the rules given above. Each rule in our calculus will correspond to a move in our game.

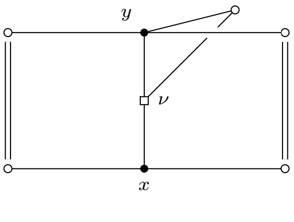
The move for parallel composition goes as follows: a player x may *fork* to turn into two new players who know the same channels as x. Graphically, if the player who plays the move is binary (i.e., they know two channels), this is pictured as:



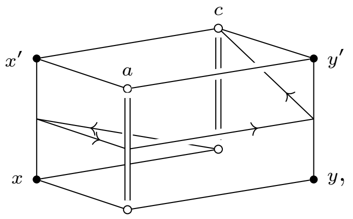
where the starting position is pictured at the bottom (it has one player x who knows two channels), the final position (with two players y_1 and y_2 who know exactly the same channels as x). There is also something in the middle denoted by π that represents the action of forking itself, but we do not delve into details here, as we only give an informal presentation. This move is in accordance with the intuition that it should model the operational description of the rule for parallel composition: according to this rule, a process $\gamma \vdash P|Q$ turns into a pair of processes $\gamma \vdash P$ and $\gamma \vdash Q$ that know exactly the same channels (i.e., among the channels in γ).

Following the same intuition, we easily define moves for the silent τ action and channel creation. In the first case, a process $\gamma \vdash \tau.P +_h R$ turns into a process $\gamma \vdash P$, i.e., a process that knows exactly the same channels. In the second case, a process $\gamma \vdash \nu a.P +_h R$ turns into a process $\gamma, a \vdash P$, i.e., a process that knows exactly the same channels, plus a new, fresh channel a. They are depicted as below left and below right respectively, again in the case where the initial player is binary.





Finally, the synchronisation rule is slightly more complex, and the move reflects the intuition of message passing: when two processes $\gamma \vdash a(b).P + R$ and $\gamma \vdash \bar{a}\langle c\rangle.Q + R'$ synchronise and turn into $\gamma \vdash P[b \mapsto c]$ and $\gamma \vdash Q$, Q sends its channel c on a and P receives it and treats it as a fresh channel. Therefore, if $a(b).P$ knows n channels among the channels of γ, $P[b \mapsto c]$ knows $n+1$ of these channels (there may be repetitions, e.g., if $a(b).P$ already knows c, then $P[b \mapsto c]$ knows c "twice"). Moreover, note that, for both processes to communicate, they need to share a common channel a. If we represent this move graphically, for a player x who knows two channels a and c and who sends c on a, and a player y who knows only one channel a, we obtain:



where the initial position is indeed again at the bottom, and in the final position, the two players have turned into x' and y', and y' knows c. Just like in the other moves, there is something in the middle that represents the action of synchronising itself: the arrow that goes from c in the bottom position to c in the top position. Again, we do not dwell on details here, but intuitively, this arrow represents the path c's name follows during the synchronisation: x first fetches c's name, then sends it on a, the message is received by y, who thus gains the knowledge of a new channel.

The congruence rule is not turned into a move in our game, as it is typically given by the fact that moves only have a local effect (i.e., when a player moves, the rest of the position is left untouched).

Stage *(i)*: positions

Let us now formalise this. All involved data will be represented as presheaves on a certain base category, say $\mathbb{C}$. Let us start with $\mathbb{C}_1$, the part of $\mathbb{C}$ which only concerns positions:

Definition 4.3.5. *Let $\mathbb{C}_1$ have*

- *an object $\star$,*
- *an object $[n]$ for each n in $\mathbb{N}$,*
- *morphisms $s_1, \ldots, s_n : \star \to [n]$ for each n in $\mathbb{N}$.*

The $\star$ object represents channels, while $[n]$ objects represent n-ary players.

Example 4.3.6. *The (informal) position of Example 4.3.4 is modelled as the presheaf X with*

- *$X(\star) = \{c_1, c_2\}$,*
- *$X([1]) = \{y_1, y_2\}$,*
- *$X([2]) = \{x\}$,*
- *$X([n]) = \varnothing$ otherwise,*

and whose action on non-trivial morphisms is:

- *$y_1 \cdot s_1 = c_1$,*
- *$y_2 \cdot s_1 = c_1$,*
- *$x \cdot s_1 = c_1$, and $x \cdot s_2 = c_2$.*

The graphical representation we have given in Example 4.3.4 is a representation of the category of elements of X (the only information lost with that representation is the order of channels, i.e., whether $x \cdot s_1$ is c_1 or c_2).

Stage *(ii)*: selecting spans

Let us now select the spans that will model moves. We will select a span for each move described above, i.e., forking π, silent action τ, channel creation ν, and synchronisation σ.

However, in order for our pseudo double category to be fibred, we need to be able to talk about plays on sub-positions. In other words, when moves involve more than one player, there should be moves that correspond to what a single player sees of the play. In our case, this means that σ, which concerns two players, should be "split" into two moves, each move corresponding to the action of a single player. It is thus split into ι, which represents *receiving* a name, and o, which represents *sending* a name.

Moreover, we ultimately want players to be able to change their behaviours according only to what they have seen of the play. We thus need to be able to define *views*, which are what a single player sees of a play. In our case, this means that π should also be split into two moves, the *left forking* move π^l and the *right forking* move π^r that correspond only to what a single avatar of the forking move sees when a player forks.

Finally, there should actually be an n-ary version of each move, since players do not have a constant arity, which means that there should be a τ_n move for each n in $\mathbb{N}$, etc. Similarly, since channels are ordered, a player receiving on their first channel is not doing the same thing as one who receives on their second one. This means that there should be an $\iota_{n,i}$ move that represents an n-ary player receiving on their ith channel for all n in $\mathbb{N}$ and i in n. Similarly, there should be an $o_{m,j,k}$ move that represents an m-ary player sending their kth channel on their jth one for all m in $\mathbb{N}$ and j, k in m. Naturally, this should be reflected in σ: there is a $\sigma_{n,i,m,j,k}$ move for all n, m in $\mathbb{N}$, i in n, and j, k in m.

Let us now choose spans for all these moves. Remember that a span $Y \xleftarrow{w} I \xrightarrow{u} X$ should represent a move whose starting position is X, whose final position is Y, and where I is the part of the position that is left untouched by the move. Following this intuition, we define the spans for all moves:

- for τ_n: $[n] \xleftarrow{[s_i]_{i\in n}} n\cdot\star \xrightarrow{[s_i]_{i\in n}} [n]$,
- for ν_n: $[n+1] \xleftarrow{[s_i]_{i\in n}} n\cdot\star \xrightarrow{[s_i]_{i\in n}} [n]$,
- for π^l_n: $[n] \xleftarrow{[s_i]_{i\in n}} n\cdot\star \xrightarrow{[s_i]_{i\in n}} [n]$,
- for π^r_n: $[n] \xleftarrow{[s_i]_{i\in n}} n\cdot\star \xrightarrow{[s_i]_{i\in n}} [n]$,
- for π_n: $[n] \mid [n] \xleftarrow{w} n\cdot\star \xrightarrow{[s_i]_{i\in n}} [n]$, where $[n] \mid [n]$ is the position defined as the pushout
$$\begin{tikzcd}
n\cdot\star \arrow[r, "{[s_i]_{i\in n}}"] \arrow[d, "{[s_i]_{i\in n}}"'] & {[n]} \arrow[d, "\mathrm{inl}"] \\
{[n]} \arrow[r, "\mathrm{inr}"'] & {[n] \mid [n],} \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{4.2}$$
and we denote by $w\colon n\cdot\star \to [n] \mid [n]$ the diagonal morphism in the pushout above,

- for $\iota_{n,i}$: $[n+1] \xleftarrow{[s_i]_{i\in n}} n \cdot * \xrightarrow{[s_i]_{i\in n}} [n]$ (notice that the player in the final position knows one more channel than the one in the initial position),
- for $o_{n,j,k}$: $[n] \xleftarrow{[s_i]_{i\in n}} n \cdot * \xrightarrow{[s_i]_{i\in n}} [n]$,
- for $\sigma_{n,i,m,j,k}$: $[n+1]\,{}_{i,n+1}|_{j,k}\,[m] \xleftarrow{w} (n+m-1)\cdot * \xrightarrow{u} [n]\,{}_i|_j\,[m]$, where $[n]\,{}_i|_j\,[m]$ is defined as the pushout below and u by universal property of pushout:

$$\begin{tikzcd}
* \arrow[rr, "inj_j"] \arrow[dr, equal] \arrow[dd, "inj_i"'] & & m\cdot * \arrow[dr, "{[s_i]_{i\in m}}"] \arrow[dd] & \\
& * \arrow[rr, "s_j"] \arrow[dd, "s_i"'] & & {[m]} \arrow[dd, "\mathrm{inr}"] \\
n\cdot * \arrow[rr] \arrow[dr, "{[s_i]_{i\in n}}"'] & & (n+m-1)\cdot * \arrow[ul, phantom, very near start, "\ulcorner"] \arrow[dr, dashed, "u"] & \\
& {[n]} \arrow[rr, "\mathrm{inl}"'] & & {[n]\,{}_i|_j\,[m],} \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd} \tag{4.3}$$

and similarly for $[n]\,{}_{i,n+1}|_{j,k}\,[m]$ and w:

$$\begin{tikzcd}
* \arrow[rr, "inj_j"] \arrow[dr, "\mathrm{inl}"] \arrow[dd, "inj_i"'] & & m\cdot * \arrow[dr, "{[s_i]_{i\in m}}"] \arrow[dd] & \\
& 2\cdot * \arrow[rr, "{[s_j,s_k]}"] \arrow[dd, "{[s_i,s_{n+1}]}"'] & & {[m]} \arrow[dd, "\mathrm{inr}"] \\
n\cdot * \arrow[rr] \arrow[dr, "{[s_i]_{i\in n}}"'] & & (n+m-1)\cdot * \arrow[ul, phantom, very near start, "\ulcorner"] \arrow[dr, dashed, "w"] & \\
& {[n+1]} \arrow[rr, "\mathrm{inl}"'] & & {[n+1]\,{}_{i,n+1}|_{j,k}\,[m].} \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd} \tag{4.4}$$

The reader may check that, if we draw the category of elements of the different presheaves defined above, we do get back the initial and final positions of the moves we informally defined before, as well as the part of those moves that is common to both positions.

Stage *(iii)*: augmenting the base category

At last, we augment our base category $\mathbb{C}_1$ with new objects and morphisms that model moves. We could here appeal to cocomma categories, as sketched in Chapter 3, which would lead to the definition below, except it would lack the morphisms $\pi^l_n, \pi^r_n \to \pi_n$ and $\iota_{n,i}, o_{m,j,k} \to \sigma_{n,i,m,j,k}$. In order to get them back, we would need to build $\mathbb{C}$ in three stages, which would make the construction slightly more complex, so we do not describe the process here, and simply give the result.

Definition 4.3.7. *Let $\mathbb{C}$ consist of $\mathbb{C}_1$, plus, for all n, m in $\mathbb{N}$, a, b in n, and c, d in m:*

- *objects π^l_n, π^r_n, π_n, ν_n, τ_n, $\iota_{n,a}$, $o_{n,a,b}$, and $\sigma_{n,a,m,c,d}$,*

- *for all v in $\{\pi^l_n, \pi^r_n, \tau_n, o_{n,a,b}\}$, morphisms $s, t : [n] \to v$,*
- *morphisms $[n+1] \xrightarrow{s} \nu_n \xleftarrow{t} [n]$ and $[n+1] \xrightarrow{s} \iota_{n,a} \xleftarrow{t} [n]$,*
- *morphisms $\pi^l_n \xrightarrow{l} \pi_n \xleftarrow{r} \pi^r_n$ and $\iota_{n,a} \xrightarrow{\rho} \sigma_{n,a,m,c,d} \xleftarrow{\varepsilon} o_{m,c,d}$,*

modulo the equations:

- *$s \circ s_i = t \circ s_i$ in $\mathbb{C}(\star, v)$, for all n in $\mathbb{N}$, i in n, a,b in n, and v in $\{\pi^l_n, \pi^r_n, \tau_n, \iota_{n,a}, o_{n,a,b}, \nu_n\}$,*
- *$l \circ t = r \circ t$ in $\mathbb{C}([n], \pi_n)$, for all n in $\mathbb{N}$,*
- *$\rho \circ t \circ s_a = \varepsilon \circ t \circ s_b$ and $\rho \circ s \circ s_{n+1} = \varepsilon \circ s \circ s_c$, in $\mathbb{C}(\star, \sigma_{n,a,m,b,c})$, for all n, m in $\mathbb{N}$, a in n, and b, c in m.*

The pseudo double category describing the π-calculus will be a sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$, entirely determined by the choice of a functor from $\mathbb{C}_{|\geq 2}$ to $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$, where $\mathbb{C}_{|\geq 2}$ denotes the full subcategory of $\mathbb{C}$ spanning the objects not in $\mathbb{C}_1$. Intuitively, this functor chooses for all new objects μ of $\mathbb{C}$ a cospan which will be thought of as the basic play consisting of just μ. This choice of cospan should of course be compatible with the choice of spans made above, as mentioned in the informal description of the process in Section 4.3.1.

In fact, for each tuple (n, m, a, c, d), $\mathbb{C}_{|\geq 2}$ locally looks like the poset

$$\begin{tikzcd}
 & & & \pi_n & & & \sigma_{n,a,m,c,d} & \\
\nu_n & \tau_n & \pi^l_n \arrow[ur, "l"] & & \pi^r_n \arrow[ul, "r"'] & \iota_{n,a} \arrow[ur, "\rho"] & & o_{m,c,d} \arrow[ul, "\varepsilon"']
\end{tikzcd}$$

viewed as a category (where we have omitted τ_m, π_m, etc for readability). We define our functor $\mathbb{C}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{C}})_H$ to map this to

$$\begin{tikzcd}
Y_{\nu_n} \arrow[d] & Y_{\tau_n} \arrow[d] & Y_{\pi^l_n} \arrow[r, "\mathrm{inl}"] \arrow[d] & Y_{\pi_n} \arrow[d] & Y_{\pi^r_n} \arrow[l, "\mathrm{inr}"'] \arrow[d] & Y_{\iota_{n,a}} \arrow[r, "\mathrm{inl}"] \arrow[d] & Y_{\sigma_{n,a,m,c,d}} \arrow[d] & Y_{o_{m,c,d}} \arrow[l, "\mathrm{inr}"'] \arrow[d] \\
\nu_n & \tau_n & \pi^l_n \arrow[r, "l"] & \pi_n & \pi^r_n \arrow[l, "r"'] & \iota_{n,a} \arrow[r, "\rho"] & \sigma_{n,a,m,c,d} & o_{m,c,d} \arrow[l, "\varepsilon"'] \\
X_{\nu_n} \arrow[u] & X_{\tau_n} \arrow[u] & X_{\pi^l_n} \arrow[u] \arrow[r, equal] & X_{\pi_n} \arrow[u] \arrow[r, equal] & X_{\pi^r_n} \arrow[u] & X_{\iota_{n,a}} \arrow[u] \arrow[r, "\mathrm{inl}"'] & X_{\sigma_{n,a,m,c,d}} \arrow[u] & X_{o_{m,c,d}} \arrow[u] \arrow[l, "\mathrm{inr}"] ,
\end{tikzcd} \tag{4.5}$$

where X_m and Y_m are the positions defined in Stage *(ii)*, and the different horizontal morphisms are all defined in (4.2), (4.3), and (4.4). It is then routine to verify that the obtained assignment is a functor.

Definition 4.3.8. *Let $\mathsf{S}_\pi : \mathbb{C}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{C}})_H$ denote the obtained functor. We call* seeds *cospans in the image of this functor.*

We now need to construct our pseudo double category based on this. Before proceeding, as we intend to provide a generic construction, we reflect a bit on the properties of our functor S_π, which leads us to the notion of signature.

4.3.2 Signatures

Required: 4.2, 4.3.1.
Recommended: $\varnothing$.

We now introduce an abstract notion of signature that generalises the functor $\mathbb{C} \to \mathsf{Cospan}(\widehat{\mathbb{C}})$ obtained above. The Steps *(i)*-*(iii)* above are just meant to be an illustration to support our definition of signature.

We begin by a few observations on the obtained functor. Our first observation is that the category $\mathbb{C}$ enjoys a natural notion of dimension: $\star$ has dimension 0, each $[n]$ has dimension 1, all τ_n, ν_n, π^l_n, π^r_n, $\iota_{n,a}$, and $o_{m,c,d}$ have dimension 2, and all π_n and $\sigma_{n,a,m,c,d}$ have dimension 3. In particular, $\mathbb{C}$ forms a *direct* category in the sense of Garner [38], i.e., it comes equipped with a functor to the ordinal ω viewed as a category, which reflects identities. Presheaves X themselves inherit a (possibly infinite) dimension: the least n such that X is empty above dimension n. The dimension of any representable is thus that of the underlying object.

Second, let us make a few additional observations on our functor $\mathsf{S}_\pi \colon \mathbb{C}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{C}})_H$:

(a) the middle object of each $\mathsf{S}_\pi(\mu)$ is y_μ;

(b) both legs of all selected cospans are monic;

(c) all morphisms between those cospans have monic components;

(d) for all such morphisms, top squares are pullbacks;

(e) finally, all initial positions X are *tight*, in the sense that all channels $c \in X(\star)$ are in the image of some $X(s_i)$.

So a first, naive notion of signature could consist of a direct category $\mathbb{C}$, equipped with a functor from $\mathbb{C}_{|\geq 2}$ to $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$ satisfying *(a)*–*(e)*. However, some of the examples we have in mind require a bit more generality, so in our abstract definition we relax things a bit. We also need to restrict it a little bit for the proof of fibredness to hold. Let us briefly explain why we need to relax the definition.

Even though, in many examples, morphisms between seeds have bottom squares that are pullback, we can see that the π-calculus does not. Indeed, consider the morphism $\rho \colon \iota_{n,a} \to \sigma_{n,a,m,c,d}$, the channel that is sent in $\sigma_{n,a,m,c,d}$ is present both in $\iota_{n,a}$ and the initial position of $\sigma_{n,a,m,c,d}$, but not in the initial position of $\iota_{n,a}$. However, we want to have a property close to that bottom square being a pullback for the proof of fibredness to hold. There are also relevant, though unpublished examples of base categories in which the top square of a morphism between seeds is not necessarily a pullback. Furthermore, we need the generated pseudo double category to accommodate morphisms of cospans whose components are non-injective. Let us thus change our tentative definition just enough to accomodate these needs. In short, we pass from injective maps to maps which are injective except perhaps on objects of dimension 0, and find an analogous generalisation for bottom squares being pullbacks. We also completely drop the requirement on top squares, as it proves useless in the definition.

Let us fix any small, direct category $\mathbb{C}$ for the rest of this section. By analogy with the base category for π, we think of objects of dimension 1 as players in a game, which may communicate with each other through objects of dimension 0. Objects of dimensions > 1 are thought of as moves in the game. Accordingly, we use the following terminology:

Terminology 4.3.9. *The* dimension *of any object of $\mathbb{C}$ is its image in ω. A* channel *is an object of dimension 0, a* player *is an object of dimension 1, and a* move *is an object of dimension > 1.*

Definition 4.3.10. *Let a natural transformation of presheaves over $\mathbb{C}$ be* 1D-injective *when all its components of dimensions > 0 are injective.*

A square in $\widehat{\mathbb{C}}$ is a 1D-pullback *when it is a pullback in all dimensions > 0.*

Notation 4.3.11. *We mark 1D-pullbacks with a dotted little square, as below*

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d] \\
C \arrow[r] & D
\end{tikzcd}$$

Definition 4.3.12. *Let $\mathbb{D}^0(\mathbb{C})$ denote the sub-pseudo double category of the pseudo double category $\mathsf{Cospan}(\widehat{\mathbb{C}})$*

- *whose horizontal category $\mathbb{D}^0(\mathbb{C})_h$ is the subcategory of $\widehat{\mathbb{C}}$ consisting of* positions*, i.e., finitely presentable presheaves of dimension ≤ 1, and 1D-injective morphisms between them,*
- *whose vertical morphisms are cospans with monic legs, and*
- *whose cells are those of $\mathsf{Cospan}(\widehat{\mathbb{C}})$ with 1D-injective components and 1D-pullback bottom squares.*

Terminology 4.3.13. *For any vertical $u{:}Y \mathrel{\bullet\!\!\!\to} X$ in $\mathbb{D}^0(\mathbb{C})_v$, X and Y are respectively called the* initial *and* final *positions of u.*

Definition 4.3.12 only makes sense because:

Proposition 4.3.14. *$\mathbb{D}^0(\mathbb{C})$ forms a sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$.*

This relies on the following direct corollary of Lemma 4.2.6:

Corollary 4.3.15. *In $\widehat{\mathbb{C}}$, for any commuting cube*

$$\begin{tikzcd}
I \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & B \arrow[dr] \arrow[dd] & \\
 & A \arrow[rr] \arrow[dd] & & C \arrow[dd, "f"] \arrow[ul, phantom, "\ulcorner", very near start] \\
I' \arrow[rr] \arrow[dr] & & B' \arrow[dr] & \\
 & A' \arrow[rr] & & C', \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

with the marked pushouts and 1D-pullback,

- *if $I' \to B'$ is 1D-injective then the front square is a 1D-pullback, and*
- *if all arrows except perhaps f are 1D-injective, then f also is.*

Proof. By pointwise application of Lemma 4.2.6. □

Proof of Proposition 4.3.14. The only non-trivial bit lies in showing that a vertical composite of componentwise 1D-injective cells with 1D-pullback bottom squares again has 1D-injective components and 1D-pullback bottom square. This is a simple consequence of Corollary 4.3.15 and the pullback lemma. □

Let us finally give a definition of tightness which generalises the one given in *(e)* for π – though the presentation differs.

Definition 4.3.16. *For all presheaves U in $\widehat{\mathbb{C}}$, let us denote by $\mathrm{pl}(U)$ the set of* players *of U, i.e., pairs (d, x) for all morphisms $x: d \to U$, where d is any representable of dimension 1. Let $\mathrm{Pl}(U)$ denote the corresponding coproduct $\sum_{(d,x)\in \mathrm{pl}(U)} d$ of representables.*

A position X is tight *if and only if the canonical morphism $\mathrm{Pl}(X) \to X$ is epi.*

Definition 4.3.17. *A* signature *consists of a small, direct category $\mathbb{C}$, together with a functor $\mathsf{S}: \mathbb{C}_{|\geq 2} \to \mathbb{D}^0(\mathbb{C})_H$ making the following square commute*

$$\begin{tikzcd} \mathbb{C}_{|\geq 2} \arrow[r, "\mathsf{S}"] \arrow[d, hook] & \mathbb{D}^0(\mathbb{C})_H \arrow[d, "\mathsf{m}"] \\ \mathbb{C} \arrow[r, hook, "\mathsf{y}"'] & \widehat{\mathbb{C}}, \end{tikzcd} \tag{4.6}$$

where m denotes the middle projection functor. We further require that for all $\mu \in \mathbb{C}_{|\geq 2}$, the initial position of $\mathsf{S}(\mu)$ be tight.

Definition 4.3.18. *Cospans in the image of S are called the* seeds *of S.*

Letting $X, Y, Z, \ldots$ range over positions, we get that any signature S maps any move M to some cospan $Y \to M \leftarrow X$ which determines its initial and final positions.

Example 4.3.19. *The functor S_π of Definition 4.3.8 is a signature. Indeed, all morphisms are evidently monic and all bottom squares are 1D-pullbacks (the only one that is not a pullback is the one that corresponds to $\rho: \iota_{n,a} \to \sigma_{n,a,m,c,d}$, and it is straightforwardly shown to be a 1D-pullback). Furthermore, all initial positions are clearly tight.*

Here is an immediate, useful consequence of the definition:

Lemma 4.3.20. *The functor S underlying any signature is fully faithful.*

Proof. Faithfulness holds by Yoneda and fullness follows from monicity of the legs of the involved cospans. □

4.3.3 From Signatures to Pseudo Double Categories

Required: 4.3.2, 2.2.9.
Recommended: 3.2.3.

We now define and give an explicit description of the pseudo double category $\mathbb{D}(\mathsf{S})$ associated to any signature S.

Let us start with the following observation:

Proposition 4.3.21. *For any pushout square of the form*

$$\begin{tikzcd}
id^{\bullet}_{Z_0} \arrow[r, "id^{\bullet}_{h}"] \arrow[d, "k"'] & id^{\bullet}_{Z} \arrow[d] \\
\mathsf{S}(\mu) \arrow[r] & P \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{4.7}$$

in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$*, if* $h \in \mathbb{D}^0(\mathbb{C})_h(Z_0, Z)$ *and* $k \in \mathbb{D}^0(\mathbb{C})_H(id^{\bullet}_{Z_0}, \mathsf{S}(\mu))$*, i.e., h is 1D-injective and k has 1D-injective components and 1D-pullback bottom square, then the whole square in fact lies in* $\mathbb{D}^0(\mathbb{C})$.

Proof. Indeed, given h and $id^{\bullet}_{Z_0} \to \mathsf{S}(\mu)$ as above, the pushout $P = (Y \to M \leftarrow X)$ always exists in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$. It is computed by taking pushouts levelwise, as in

$$\begin{tikzcd}
 & Y_0 \arrow[rr] \arrow[d] & & Y \arrow[d, dashed] \\
 & \mu \arrow[rr] & & M \\
Z_0 \arrow[ur] \arrow[uur] \arrow[rr] \arrow[dr] & & Z \arrow[uur] \arrow[ur] \arrow[dr] & \\
 & X_0 \arrow[rr] \arrow[uu] & & X, \arrow[uu, dashed]
\end{tikzcd} \tag{4.8}$$

where $\mathsf{S}(\mu) = (Y_0 \to \mu \leftarrow X_0)$ and the dashed arrows are obtained by universal property of pushout. Now, monos are stable under pushouts in Set and colimits are pointwise in presheaf categories, so 1D-injectivity of all components follows from 1D-injectivity of all involved morphisms. Finally, both bottom squares are pullbacks, hence 1D-pullbacks as desired, either by Lemma 4.2.1 or by the pushout lemma and Lemma 4.2.4. □

Remark. *In the proposition above, k necessarily has 1D-pullback bottom square. Indeed, both its top and bottom squares are pullback by Lemma 4.2.1.*

Definition 4.3.22. *A* move *is any cospan M isomorphic to one obtained as a pushout of the form* (4.7).

Definition 4.3.23. *The pseudo double category* $\mathbb{D}(\mathsf{S})$ *associated to any signature* S *is the smallest sub-pseudo double category of* $\mathbb{D}^0(\mathbb{C})$ *such that*

- $\mathbb{D}(\mathsf{S})_h$ *is* $\mathbb{D}^0(\mathbb{C})_h$*;*
- $\mathbb{D}(\mathsf{S})_H$ *is replete and contains all moves;*
- $\mathbb{D}(\mathsf{S})$ *is* locally full*, i.e., if a cell of* $\mathbb{D}^0(\mathbb{C})$ *has its perimeter in* $\mathbb{D}(\mathsf{S})$*, then it is in* $\mathbb{D}(\mathsf{S})$.

Remark. *By Proposition 4.3.21, saying that* $\mathbb{D}(\mathsf{S})$ *contains all moves entails that all pushout squares defining moves lie inside* $\mathbb{D}(\mathsf{S})$*.*

That $\mathbb{D}(\mathsf{S})$ is well-defined is easy: it is the intersection of all sub-pseudo double categories of $\mathbb{D}^0(\mathbb{C})$ that verify all three points above, and $\mathbb{D}^0(\mathbb{C})$ is obviously one such pseudo double category, so we are taking the intersection of a non-empty family.

It is still useful to give a concrete description of $\mathbb{D}(\mathsf{S})$. First, its horizontal category is just $\mathbb{D}^0(\mathbb{C})_h$. Regarding vertical morphisms, $\mathbb{D}(\mathsf{S})$ must contain all moves, and since it is stable under vertical composition, it must also contain all finite composites of moves. By repleteness, it should also contain all vertical morphisms isomorphic to such vertical composites. We thus define:

Definition 4.3.24. *A* play *is any vertical morphism isomorphic to some vertical composite of moves.*

Proposition 4.3.25. $\mathbb{D}(\mathsf{S})$ *is precisely the locally full sub-pseudo double category of* $\mathbb{D}^0(\mathbb{C})$ *obtained by restricting vertical morphisms to plays.*

Proof. By construction, it is enough to show that the given data forms a sub-pseudo double category of $\mathsf{Cospan}(\widehat{\mathbb{C}})$, which is easy. □

4.3.4 Fibredness and Categories of Plays

Required: 4.3.3.
Recommended: 3.2.4.

For a given pseudo double category $\mathbb{D}$, the categories of plays studied in previous work [50, 29] come in several flavours. A first variant is based on the following category:

Definition 4.3.26. *Let* $\mathbb{E}$ *denote the category*

- *whose objects are vertical morphisms of* $\mathbb{D}$*,*
- *and whose morphisms* $u \to u'$ *are pairs* (w, α) *as below left, considered equivalent up to the equivalence relation generated by equating* (w, α) *with* $(w', \alpha \circ (u \bullet \gamma))$*, for all cells* γ *as below right:*

$$\begin{tikzcd}
T \arrow[r, "s"] \arrow[d, "w"', "\bullet" marking] & Z' \arrow[dd, "u'", "\bullet" marking] \\
Z \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] \arrow[d, "u"', "\bullet" marking] & {} \\
Y \arrow[r, "r"] & Y'
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
T' \arrow[r] \arrow[dr, "w'"', "\bullet" marking] & T \arrow[r, "s"] \arrow[d, "w", "\bullet" marking] & Z' \arrow[dd, "u'", "\bullet" marking] \\
{} \arrow[ur, phantom, "\overset{\gamma}{\Rightarrow}"] & Z \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] \arrow[d, "u"', "\bullet" marking] & {} \\
 & Y \arrow[r, "r"'] & Y'.
\end{tikzcd}
\tag{4.9}$$

Notation 4.3.27. *We denote the involved equivalence relation by* $\sim$*. Furthermore, in principle,* $\mathbb{E}$ *depends on* $\mathbb{D}$*, which should appear in the notation. For readability, we will rely on context to disambiguate.*

In order to define composition in this category, one needs to consider all diagrams of shape the solid part of

$$\begin{tikzcd}
Z'' \arrow[r, dashed] \arrow[d, dashed, "w''\bullet"'] \arrow[dr, phantom, "\overset{\gamma}{\Rightarrow}"] & Z' \arrow[r, "s'"] \arrow[d, "w'\bullet"'] & Y'' \arrow[ddd, "\bullet u''"] \\
Z \arrow[r, "s"] \arrow[d, "w\bullet"'] & Y' \arrow[dd, "\bullet u'"] \arrow[r, phantom, "\overset{\beta}{\Rightarrow}"] & \\
Y \arrow[d, "u\bullet"'] \arrow[r, phantom, "\overset{\alpha}{\Rightarrow}"] & & \\
X \arrow[r, "r"'] & X' \arrow[r, "r'"'] & X''.
\end{tikzcd}$$

Fibredness then comes in by requiring the existence of a cell γ as shown, which is canonical in a certain sense. This allows us to define the composite of (w, α) and (w', β) as the equivalence class of $(w \bullet w'', \beta \circ (\alpha \bullet \gamma))$.

Here is the long-awaited fibredness property, which uses the notion of fibration (see Section 2.2.2):

Definition 4.3.28. *A pseudo double category* $\mathbb{D}$ *is* fibred *if and only if the functor* $\mathrm{cod}{:}\, \mathbb{D}_H \to \mathbb{D}_h$ *is a fibration.*

Fibredness is related to Grandis and Paré's double categorical Kan extensions [44] and to Shulman's *framed* bicategories [94].

Proposition 4.3.29. *If* $\mathbb{D}$ *is fibred, then* $\mathbb{E}$ *is indeed a category.*

The category of plays $\mathbb{E}_X$ over any position X used in [50, 29] is then obtained as the comma category of $\mathbb{E}$ over the functor $\ulcorner X \urcorner : 1 \to \mathbb{D}_h$ picking X. In Chapter 5, to study views and plays in HON games, we will use another variant:

Definition 4.3.30. *Let* $\mathbb{E}(X)$ *denote the fibre of* $\mathbb{E}$ *over* X.

Explicitly, objects of $\mathbb{E}(X)$ are plays $u{:}Y \overset{\bullet}{\to} X$, and morphisms are those of $\mathbb{E}$, as on the left in (4.9), which have id_X as their lower border.

Our next goal is to give an abstract framework to show that, given some signature S, $\mathbb{D}(\mathsf{S})$ is fibred. We want this framework to be abstract to be able to use it on the existing examples of CCS and the π-calculus, but also on a signature for HON games in order to relate the categories $\mathbb{E}(X)$ for HON games to the categories of views and plays given by Tsukada and Ong [97].

4.4 Fibredness

In this section, we study the fibredness property abstractly. In order to do that, we first define a candidate cartesian lifting of any morphism in $\mathbb{D}(\mathsf{S})_h$ using factorisation systems and show that it has all the good properties of cartesian liftings, except that it is unclear whether it lies inside $\mathbb{D}(\mathsf{S})_H$, which is proved in the rest of the section. We then recast the definitions of 1D-injectivity and 1D-pullbacks in terms closer to the defining properties of injective maps and pullbacks. We then give a necessary and sufficient condition for our candidate lifting to lie in $\mathbb{D}(\mathsf{S})_H$, hence for $\mathbb{D}(\mathsf{S})$ to be fibred. However, this condition is not very useful in practice, so, in the last part, we give a sufficient condition for $\mathbb{D}(\mathsf{S})$ to be fibred that is easier to verify.

4.4.1 Fibredness through Factorisation Systems

Required: 4.3.4, 2.2.8.
Recommended: $\varnothing$.

Our main tool to prove that the pseudo double category $\mathbb{D}(\mathsf{S})$ generated by a signature S is fibred will be cofibrantly generated factorisation systems (see Section 2.2.8). Let us explain the core idea of the proof. In the setting of Example 2.2.93, any factorisation system $(\mathcal{L},\mathcal{R})$ on $\mathcal{C}$ yields a fibred sub-pseudo double category of $\mathsf{Cospan}(\mathcal{C})$.

We are going to refine the pseudo double category $\mathsf{Cospan}(\mathcal{C})$ into a fibred pseudo double category. Let us consider $\mathsf{Cospan}_{\mathcal{L},\mathcal{R}}(\mathcal{C})$, the locally full sub-pseudo double category of $\mathsf{Cospan}(\mathcal{C})$

- whose horizontal category is $\mathcal{C}$,
- and whose vertical morphisms $Y \relbar\mkern-9mu\bullet\mkern-9mu\rightarrow X$ are cospans $Y \xrightarrow{s} U \xleftarrow{l} X$ with $l \in \mathcal{L}$.

Proposition 4.4.1. *The pseudo double category* $\mathsf{Cospan}_{\mathcal{L},\mathcal{R}}(\mathcal{C})$ *forms a sub-pseudo double category of* $\mathsf{Cospan}(\mathcal{C})$ *that is fibred if* $\mathcal{C}$ *has pullbacks.*

Proof. That $\mathsf{Cospan}_{\mathcal{L},\mathcal{R}}(\mathcal{C})$ forms a pseudo double category is a simple consequence of Lemma 2.2.92. To see that it is fibred, consider any vertical morphism $Y \xrightarrow{f} U \xleftarrow{l} X$ and horizontal morphism $X' \xrightarrow{h} X$. In order to construct a cartesian lifting of (f,l) along h, we factor the composite $X' \xrightarrow{h} X \xrightarrow{l} U$ as $X' \xrightarrow{l'} U' \xrightarrow{h'} U$, with $l' \in \mathcal{L}$ and $h' \in \mathcal{R}$, and then take the pullback of f and h', as in the front face below:

$$\begin{tikzcd}
Y'' \arrow[drr, bend left, "q''"] \arrow[dr, dashed, "s''"'] \arrow[dd, "f''"'] & & \\
& Y' \arrow[r, "h''"'] \arrow[dd, "f'"'] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[dd, "f"] \\
U'' \arrow[drr, bend left, "q'"] \arrow[dr, dashed, "s'"'] & & \\
& U' \arrow[r, "h'"] & U \\
X'' \arrow[uu, "l''"] \arrow[drr, bend left, "q"] \arrow[dr, "s"'] & & \\
& X' \arrow[r, "h"] \arrow[uu, "l'"] & X. \arrow[uu, "l"']
\end{tikzcd} \tag{4.10}$$

The obtained morphism (h,h',h'') is generally not cartesian in $\mathsf{Cospan}(\mathcal{C})_H$, but let us show that it is in $\mathsf{Cospan}_{\mathcal{L},\mathcal{R}}(\mathcal{C})$. For this, consider any morphism (q,q',q'') to U such that $q = hs$ as above; then since $l'' \in \mathcal{L}$ (by hypothesis) and $h' \in \mathcal{R}$ (by construction), we obtain by the lifting property a unique s' making everything in sight commute. But then the universal property of pullback gives the desired s''. □

For any signature S over some base category $\mathbb{C}$, we will try to apply this construction to the pseudo double category of plays $\mathbb{D}(\mathsf{S})$ over S, with the factorisation system cofibrantly generated (remember Theorem 2.2.90) by the set

J_{S} of all "t-legs", i.e., the set of morphisms $X \xrightarrow{t} M$ for $Y \xrightarrow{s} M \xleftarrow{t} X$ spanning seeds. A map is then in $J_{\mathsf{S}}^{\perp}$ when no new move is added "forwards", i.e., following the direction of time. Indeed, recalling that each M occurring in a seed should be representable, giving a square

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "t"'] & U \arrow[d, "r"] \\
M \arrow[r, "\mu"'] & V
\end{tikzcd}$$

amounts by Yoneda to picking a move μ in V, whose initial position X is already available in U. The map r is then in $\mathcal{R}$ when all such moves are also already in U.

Our goal now reduces to showing that $\mathbb{D}(\mathsf{S})$ is fibred as a sub-pseudo double category of $\mathsf{Cospan}_{{}^{\perp}(J_{\mathsf{S}}^{\perp}),J_{\mathsf{S}}^{\perp}}(\widehat{\mathbb{C}})$ – which we henceforth abbreviate to $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$. The difficulty is that, in a situation like (4.10), the factorisation system yields a cartesian lifting (h, h', h'') in $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$, of which we will further need to prove that (1) it lies in $\mathbb{D}(\mathsf{S})_H$, and (2) it is also cartesian there. Point (2) reduces to proving that if (q, q', q'') is in $\mathbb{D}(\mathsf{S})_H$ then so is (s, s', s'').

In fact, assuming that the candidate lifting is in $\mathbb{D}(\mathsf{S})$, its cartesianness follows from the fact that all mediating arrows, computed as in (4.10), are also in $\mathbb{D}(\mathsf{S})$. Indeed, we have:

Lemma 4.4.2. *$\mathbb{D}(\mathsf{S})_H$ has the left cancellation property: for all β and α in $\mathsf{Cospan}(\widehat{\mathbb{C}})$ such that $\beta \circ \alpha$ and β are in $\mathbb{D}(\mathsf{S})_H$, then also α is in $\mathbb{D}(\mathsf{S})_H$.*

Proof. By 1D-analogues of the pullback lemma and left cancellation for monomorphisms. □

It thus remains to prove that the candidate lifting is a play, and that the morphism (h, h', h'') lies in $\mathbb{D}(\mathsf{S})_H$. Let us record this as:

Lemma 4.4.3. *Assume that in all situations like* (4.10)*, if U is a play and h is 1D-injective, then U' is again a play and (h, h', h'') is in $\mathbb{D}(\mathsf{S})_H$. Then, $\mathbb{D}(\mathsf{S})$ is fibred.*

We thus consider conditions for this to hold. In Section 4.4.3, we show that if it holds for seeds, then it extends to all plays. In Section 4.4.4, we investigate conditions for the result to hold for seeds.

This all relies on a few elementary facts about 1D-pullbacks and 1D-injective maps in presheaf categories, which we now prove.

4.4.2 A Little Theory of 1D-Pullbacks and 1D-Injectivity

Required: 2.2.5.
Recommended: ∅.

Let us start by recasting the definitions of 1D-injectivity and 1D-pullback in the following setting:

Definition 4.4.4. *A* one-way category *consists of category $\mathbb{C}$ equipped with a functor to 2, the ordinal 2 viewed as a category.*

Any direct category $d\colon \mathbb{C} \to \omega$ may be viewed as a one-way category by postcomposing with the functor $\omega \to 2$ mapping everyone to 1 except 0 which is mapped to itself.

Definition 4.4.5. *Let $d\colon \mathbb{C} \to 2$ be a one-way category. The* dimension *of an object of $\mathbb{C}$ is its image by d. A natural transformation between presheaves over $\mathbb{C}$ is* 1D-injective *if and only if its components on objects of dimension 1 are injective. A square of natural transformations is a 1D-pullback if and only if it is at all objects of dimension 1.*

Proposition 4.4.6. *The definitions of dimension (or, more precisely, whether a dimension is equal to 0 or not), 1D-injectivity, and 1D-pullbacks given for direct categories $d\colon \mathbb{C} \to \omega$ coincide with their analogues for the corresponding one-way category $\mathbb{C} \to \omega \to 2$.*

Proof. Trivial. □

We here work in the simpler setting of presheaves over a one-way category, but by the proposition we may transport our results from the one-way categorical setting to the direct categorical one. We will do so silently in the sequel.

There are several functors from one-way categories to categories, but the important one for us restricts its argument to dimension 1:

Definition 4.4.7. *Let $\pi_1 \colon \mathsf{Cat}/2 \to \mathsf{Cat}$ denote pullback along $\ulcorner 1 \urcorner \colon 1 \to 2$.*

In principle, this should rely on some global choice of pullbacks, but the easiest is to pick the pullbacks making each arrow $\mathsf{i}_1 \colon \pi_1(\mathbb{C}) \hookrightarrow \mathbb{C}$ an inclusion.

Notation 4.4.8. *We denote $\pi_1(\mathbb{C})$ by $\mathbb{C}_{|1}$.*

We now have the standard chain of adjunctions:

$$\begin{tikzcd} \widehat{\mathbb{C}_{|1}} \arrow[r, bend left=40, "\Sigma_{\mathsf{i}_1}"] \arrow[r, bend right=40, "\Pi_{\mathsf{i}_1}"'] \arrow[r, phantom, shift left=3, "\perp"] \arrow[r, phantom, shift right=3, "\perp"] & \widehat{\mathbb{C}}, \arrow[l, "\Delta_{\mathsf{i}_1}" description] \end{tikzcd}$$

where Δ_{i_1}, $\sum_{\mathsf{i}_1}$ and $\prod_{\mathsf{i}_1}$ respectively denote restriction, left Kan extension and right Kan extension along the opposite of i_1.

Proposition 4.4.9. *A morphism in $\widehat{\mathbb{C}}$ is 1D-injective if and only if its image by Δ_{i_1} is injective.*

A square in $\widehat{\mathbb{C}}$ is a 1D-pullback if and only if its image by Δ_{i_1} is a pullback.

Proof. By definition and the fact that limits are pointwise in presheaf categories. □

It is instructive to push things just a bit further. In particular, we establish a characterisation of 1D-injectivity and pullbacks analogous to the standard universal properties of injectivity and pullbacks, though relative to objects of $\widehat{\mathbb{C}_{|1}}$. Our first step is:

Proposition 4.4.10. *The left adjoint $\sum_{\mathsf{i}_1}$ is full and faithful, and the comonad $\sum_{\mathsf{i}_1} \circ \Delta_{\mathsf{i}_1}$ is idempotent, so that $\widehat{\mathbb{C}_{|1}}$ is a coreflective, full subcategory of $\widehat{\mathbb{C}}$.*

Proof. It is well known [60, Proposition 4.23] that the unit of the adjunction is an isomorphism when we extend and restrict along a fully-faithful functor. Furthermore, it is also well-known [76] that if the unit of an adjunction is an isomorphism, then the left adjoint is full and faithful. The comultiplication of the induced comonad is then an isomorphism by construction. □

Our characterisations will stem from the more general:

Lemma 4.4.11. *Consider any full coreflection* $L\colon \mathbb{C} \underset{\perp}{\rightleftarrows} \mathbb{D} :R$.

For any small category J and functor $D\colon J \to \mathbb{D}$, if RD has a limit in $\mathbb{C}$, then $L(\lim_j RD(j))$ has the universal property of a limit of D relative to objects of $\mathbb{C}$, i.e., we have for all $X \in \mathbb{C}$:

$$\int_{j\in J} \mathbb{D}(LX, D(j)) \cong \mathbb{D}(LX, L(\lim_j RD(j)))$$

naturally in X.

Proof. We have

$$\begin{aligned}\int_{j\in J} \mathbb{D}(LX, D(j)) \cong \int_{j\in J} \mathbb{C}(X, RD(j)) &\cong \mathbb{C}(X, \lim_j RD(j)) \\ &\cong \mathbb{D}(LX, L(\lim_j RD(j))),\end{aligned}$$

where the last step is by full faithfulness of L. □

Corollary 4.4.12. *A square in $\widehat{\mathbb{C}}$ as below left is a 1D-pullback if and only if for all $X \in \widehat{\mathbb{C}_{|1}}$, u and v as below right making the outer diagram commute, there is a unique mediating morphism h as shown, such that $ph = u$ and $qh = v$:*

$$\begin{tikzcd} A \arrow[r, "q"] \arrow[d, "p"'] & C \arrow[d, "g"] \\ B \arrow[r, "f"'] & D \end{tikzcd} \qquad \begin{tikzcd} \Sigma_{\mathsf{i}_1}(X) \arrow[rd, dashed, "h"] \arrow[rrd, bend left, "v"] \arrow[rdd, bend right, "u"'] & & \\ & A \arrow[r, "q"] \arrow[d, "p"'] & C \arrow[d, "g"] \\ & B \arrow[r, "f"'] & D. \end{tikzcd}$$

Proposition 4.4.13. *Consider any morphism $m\colon X \to Y$ in $\widehat{\mathbb{C}}$. The following are equivalent:*

1. *m is 1D-injective;*
2. *for all $f, g\colon \Sigma_{\mathsf{i}_1}(Z) \to X$, $mf = mg$ implies $f = g$;*
3. *the square*

$$\begin{tikzcd} X \arrow[r, equal] \arrow[d, equal] & X \arrow[d, "m"] \\ X \arrow[r, "m"'] & Y \end{tikzcd} \tag{4.11}$$

is a 1D-pullback.

Proof. By definition, these three items are respectively equivalent to

- *(1') $\Delta_{\mathsf{i}_1}(m)$ is monic,*
- *(2') for all $f, g\colon Z \to \Delta_{\mathsf{i}_1}(X)$, $\Delta_{\mathsf{i}_1}(m)f = \Delta_{\mathsf{i}_1}(m)g$ implies $f = g$;*
- *(3') the image by Δ_{i_1} of the square* (4.11) *is a pullback.*

But now these three are well-known to be equivalent. □

4.4.3 A Necessary and Sufficient Fibredness Criterion

Required: 4.4.1, 4.4.2.
Recommended: ∅.

We now prove some basic facts about $\mathsf{Cospan}(\widehat{\mathbb{C}})$, $\mathbb{D}^0(\mathbb{C})$, and $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$, from which we derive useful results about plays, and eventually our main abstract result, namely that, under the hypothesis that seeds admit cartesian restrictions (which we investigate independently in the next section), $\mathbb{D}(\mathsf{S})$ is fibred.

Let us start with some notation:

Notation 4.4.14. *The cospan underlying any play $u\colon Y \mathrel{\to\mkern-14mu\bullet\mkern-6mu\to} X$ will be denoted by $Y \xrightarrow{s_u} U \xleftarrow{t_u} X$ (using capitalisation for the middle object). We will often denote cospans $Y \xrightarrow{s} U \xleftarrow{t} X$ simply by $\langle U \rangle$, leaving the context provide the missing information. Furthermore, pushouts exist in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$ (they are given by taking pushouts levelwise) and when we write pushouts of cospans, the notation means that they are cospans in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$. We will often take pushouts of cospans in $\mathbb{D}^0(\mathbb{C})_H$, but such pushouts are not necessarily pushouts in $\mathbb{D}^0(\mathbb{C})_H$ (the square itself does not necessarily lie in $\mathbb{D}^0(\mathbb{C})_H$).*

Let us start with some preliminary work about tightness.

Lemma 4.4.15. *For any position X, we have $\mathrm{Pl}(X) \cong \sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(X))$ (recalling Definition 4.3.16). In particular, X is tight if and only if $\varepsilon_X\colon \sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(X)) \to X$ is epi.*

Proof. By definition. □

Lemma 4.4.16. *Consider any diagram*

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "k"] \\
C \arrow[r, "h"'] \arrow[ur, "l"] & D
\end{tikzcd}$$

in $\widehat{\mathbb{C}}$ where only the outer square and the bottom right triangle are known to commute, i.e., $kf = hg$ and $kl = h$. If A is tight and k is 1D-injective, then also the top left triangle commutes.

Proof. Post-composing with k, we have by hypothesis that $kf = hg = klg$, hence $kf\varepsilon_A = klg\varepsilon_A$. By 1D-injectivity of k and Proposition 4.4.13, we get $f\varepsilon_A = lg\varepsilon_A$. By tightness of A and Lemma 4.4.15, we finally obtain $f = lg$. □

Definition 4.4.17. *The* cofree invariant position *of a cospan $Y \to U \leftarrow X$ is given by the pullback*

$$\begin{tikzcd}
Z \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & Y \arrow[d] \\
X \arrow[r] & U.
\end{tikzcd}$$

The terminology is justified by the following:

Proposition 4.4.18. *Fixing any global choice of pullbacks, taking the cofree invariant position Z_u of any play u induces a functor $Z_{-}\colon \mathsf{Cospan}(\widehat{\mathbb{C}})_H \to \widehat{\mathbb{C}_{|1}}$ which is right adjoint to the subcategory inclusion $\widehat{\mathbb{C}_{|1}} \hookrightarrow \mathsf{Cospan}(\widehat{\mathbb{C}})_H$. The inclusion being obviously full and faithful, the unit is an isomorphism, and the associated comonad is idempotent.*

Proof. Let us take a play $\langle U \rangle$, denote by Z its cofree invariant positions, take a position T and a morphism $id^{\bullet}_T \to \langle U \rangle$, i.e., a diagram as the solid part of

$$\begin{tikzcd}
 & T \arrow[rr] \arrow[d, equal] & & Y \arrow[d] \\
 & T \arrow[rr] \arrow[dd, equal] & & U \\
T \arrow[uur, equal] \arrow[dr, equal] \arrow[rr, dashed] & & Z \arrow[uur] \arrow[dr] \arrow[ur, phantom, very near start, "\lrcorner"] & \\
 & T \arrow[rr] & & X, \arrow[uu]
\end{tikzcd}$$

from which we get a dashed morphism $T \to Z$ by universal property of pullback. Conversely, given a morphism $T \to Z$, the morphism $id^{\bullet}_T \to \langle U \rangle$ is simply built by composition. □

Lemma 4.4.19. *Any pushout in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$ as below left, where Z is a position, may be factored as below right, where Z_0 is the cofree invariant position of U:*

$$\begin{tikzcd}
id^{\bullet}_Z \arrow[r] \arrow[d] & \langle U \rangle \arrow[d] \\
\langle V \rangle \arrow[r] & \langle W \rangle \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}
\qquad
\begin{tikzcd}
id^{\bullet}_Z \arrow[r] \arrow[d] & id^{\bullet}_{Z_0} \arrow[r] \arrow[d] & \langle U \rangle \arrow[d] \\
\langle V \rangle \arrow[r] & \langle V' \rangle \arrow[r] \arrow[ul, phantom, very near start, "\ulcorner"] & \langle W \rangle. \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

If $\langle V \rangle$ is isomorphic to $id^{\bullet}_{Z'}$ for some position Z', then $\langle V' \rangle$ is isomorphic to $id^{\bullet}_{Z'_0}$ for some position Z'_0.

Proof. The last point is a consequence of colimits being pointwise in presheaf categories. For the first, we get a map $Z \to Z_0$ such that $id^{\bullet}_Z \to \langle U \rangle = id^{\bullet}_Z \to id^{\bullet}_{Z_0} \to \langle U \rangle$ by adjunction. We can then define $\langle V' \rangle$ as the pushout of $\langle V \rangle$ along $id^{\bullet}_Z \to id^{\bullet}_{Z_0}$ and obtain a unique morphism $\langle V' \rangle \to \langle W \rangle$ by its universal property: this yields a diagram as desired, whose right-hand square is again a pushout by the pushout lemma. □

Lemma 4.4.20. *Cofree invariant positions are stable under pushout in the following sense: if Z is the cofree invariant position of $\langle U \rangle$ and*

$$\begin{tikzcd}
id^{\bullet}_{Z} \arrow[r] \arrow[d] & id^{\bullet}_{Z'} \arrow[d] \\
\langle U \rangle \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & \langle U' \rangle
\end{tikzcd}$$

is a pushout, then Z' is the cofree invariant position of $\langle U' \rangle$.

Proof. Let us first name the involved presheaves: $\langle U \rangle = (Y \to U \leftarrow X)$ and $\langle U' \rangle = (Y' \to U' \leftarrow X')$. Since Z is the cofree invariant position of $\langle U \rangle$, we may apply Corollary 4.2.7 to

$$\begin{tikzcd}
Z \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & Y \arrow[dr] \arrow[dd] & \\
& Z' \arrow[rr] \arrow[dd] \arrow[ur, phantom, "\ulcorner", very near end, shift right=2] & & Y' \arrow[dd] \\
X \arrow[rr, hook] \arrow[dr] & & U \arrow[dr] & \\
& X' \arrow[rr] \arrow[ur, phantom, "\ulcorner", very near end, shift right=2] & & U',
\end{tikzcd}$$

to obtain that the front face is also a pullback. □

Lemma 4.4.21. *In* $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$*, pushout squares are preserved by vertical composition. More explicitly, given two vertically composable pushouts as below left and centre, the composite square below right is again a pushout:*

$$\begin{tikzcd}
\langle U_0 \rangle \arrow[r] \arrow[d] & \langle U_1 \rangle \arrow[d] & \langle V_0 \rangle \arrow[r] \arrow[d] & \langle V_1 \rangle \arrow[d] & \langle U_0 \rangle \bullet \langle V_0 \rangle \arrow[r] \arrow[d] & \langle U_1 \rangle \bullet \langle V_1 \rangle \arrow[d] \\
\langle U_2 \rangle \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & \langle U \rangle & \langle V_2 \rangle \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & \langle V \rangle & \langle U_2 \rangle \bullet \langle V_2 \rangle \arrow[r] \arrow[ur, phantom, "\ulcorner", very near end] & \langle U \rangle \bullet \langle V \rangle.
\end{tikzcd}$$

Proof. Let us first name the involved presheaves: $\langle U_i \rangle = (Y_i \to U_i \leftarrow X_i)$, $\langle V_i \rangle = (Z_i \to V_i \leftarrow Y_i)$, and similarly for $\langle U \rangle$ and $\langle V \rangle$. If we call Λ the posetal category with objects 0, 1, and -1, and morphisms generated by $0 \to 1$ and $0 \to -1$ (the "walking span" category), we introduce a bifunctor from $\Lambda \times \Lambda$ to $\widehat{\mathbb{C}}$ through the following diagram:

$$\begin{tikzcd}
V_1 & V_0 \arrow[l] \arrow[r] & V_2 \\
Y_1 \arrow[u] \arrow[d] & Y_0 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & Y_2 \arrow[u] \arrow[d] \\
U_1 & U_0 \arrow[l] \arrow[r] & U_2.
\end{tikzcd}$$

By computing its colimit first horizontally, then vertically, we get $\langle U \rangle \bullet \langle V \rangle$, which by interchange of colimits is the desired pushout. □

Lemma 4.4.22. *Any morphism in* $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})_H$ *is cartesian if and only if it has the shape of the front face of* (4.10)*, i.e., its top square is a pullback and its middle morphism is in* $J_{\mathsf{S}}^{\perp}$*.*

Proof. The "if" direction follows from the proof of Proposition 4.4.1. For the "only if" direction, the considered properties are stable under composition with isomorphisms in $\mathsf{Cospan}_{J_\mathsf{S}}(\widehat{\mathbb{C}})_H$. But any cartesian $\alpha\colon\langle U\rangle\to\langle U'\rangle$ is uniquely isomorphic in $\mathsf{Cospan}_{J_\mathsf{S}}(\widehat{\mathbb{C}})_H/\langle U'\rangle$ to the cartesian lifting of $\langle U'\rangle$ along $\mathrm{cod}(\alpha)$ computed as in (4.10), hence the result. □

When we want to show that a morphism of cospans is cartesian, we need by Lemma 4.4.22 to show that the middle morphism is in $J_\mathsf{S}^\perp$. The next few lemmas prepare for Corollary 4.4.25, which is the main tool we will use to show that some morphisms belong to $J_\mathsf{S}^\perp$.

Some 1D-pullbacks in $\widehat{\mathbb{C}}$ satisfy the universal property of pullbacks with respect to tight positions. Concretely:

Lemma 4.4.23. *For any commuting diagram as the solid part of*

$$\begin{tikzcd}
T \arrow[rrd, bend left, "h"] \arrow[rd, dashed, "k"] \arrow[rdd, bend right] & & \\
& A \arrow[r, "f"] \arrow[d, hook] \arrow[rd, phantom, "\lrcorner", dotted] & C \arrow[d] \\
& B \arrow[r] & D
\end{tikzcd}$$

with the marked mono and 1D-pullback, and where T is a tight position, there exists a unique map k making the diagram commute.

Proof. We construct in turn both dashed maps in

$$\begin{tikzcd}
\mathrm{Pl}(T) \arrow[r, dashed, "l"] \arrow[d, two heads, "\varepsilon_T"'] & A \arrow[r, "f"] \arrow[d, hook] \arrow[rd, phantom, "\lrcorner", dotted] & C \arrow[d] \\
T \arrow[ru, dashed, "k"] \arrow[rru, "h"'] \arrow[r] & B \arrow[r] & D :
\end{tikzcd}$$

- l follows from universal property of 1D-pullbacks (Corollary 4.4.12), since $\mathrm{Pl}(T)\cong\sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(T))$ (Lemma 4.4.15),
- k then follows again by Lemma 4.4.15 (which ensures that ε_T is epi) and lifting in the (Epi, Mono) factorisation system.

The construction of k however does not *a priori* ensure that $f\circ k=h$, but $f\circ k\circ\varepsilon_T=f\circ l=h\circ\varepsilon_T$ which entails the result by tightness. □

This yields an analogue of the pullback lemma:

Lemma 4.4.24. *In any commuting diagram*

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[rd, phantom, very near start, "\lrcorner"] & B \arrow[r, hook] \arrow[d] \arrow[rd, phantom, "\lrcorner", dotted] & C \arrow[d] \\
D \arrow[r] & E \arrow[r] & F
\end{tikzcd}$$

with the marked mono, pullback and 1D-pullback, the outer rectangle has the universal property of pullbacks w.r.t. tight positions.

Proof. A diagram chasing similar to the proof of the pullback lemma, using Lemma 4.4.23. □

Corollary 4.4.25. *For any seeds $Y \to M \leftarrow X$ and $S \to C \leftarrow T$, any commuting diagram as the solid part of*

$$\begin{tikzcd}
T \arrow[r, dashed] \arrow[d] \arrow[dr] & U \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\
C \arrow[rr, bend right] & V \arrow[r] & M
\end{tikzcd}$$

with the marked 1D-pullback, such that at least one of $U \to X$ and $U \to V$ is monic, may be completed as shown.

Proof. Because M is a seed, the bottom square of $\mathsf{S}(C \to M)$ yields a morphism $T \to X$ making the diagram commute. We conclude by Lemma 4.4.23. □

Lemma 4.4.26. *For any commuting diagram of 1D-injective maps of the form*

$$\begin{tikzcd}
U \arrow[r, hook, "s_U"] \arrow[d, "f_U"'] \arrow[dr, phantom, "\lrcorner", very near start] & W \arrow[d, "f"'] & V \arrow[l, hook', "s_V"'] \arrow[d, "f_V"] \arrow[dl, phantom, "\llcorner", very near start] \\
U' \arrow[r, hook, "s_{U'}"'] & W' & V' \arrow[l, hook', "s_{V'}"]
\end{tikzcd}$$

with the marked 1D-pullbacks in $\widehat{\mathbb{C}}$, such that

- *s_U and s_V are jointly surjective and both monic,*
- *$s_{U'}$ and $s_{V'}$ are jointly surjective and both monic,*
- *f_U and f_V are in $J_{\mathsf{S}}^{\perp}$,*

we have that $f \in J_{\mathsf{S}}^{\perp}$.

Proof. Consider any morphism $T \to C$ in J_{S} and commuting square

$$\begin{tikzcd}
T \arrow[r] \arrow[d] & W \arrow[d] \\
C \arrow[r] & W'.
\end{tikzcd}$$

We want to show that there is a unique diagonal filler $C \to W$. By Proposition 4.4.13, uniqueness follows from 1D-injectivity of f and the fact that $C \cong \sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(C))$, so we only need to show existence. Furthermore, by joint surjectivity and because C is a representable, $C \to W'$ factors either through U' or through V' (possibly both).

Both cases being symmetric, we only treat one. If $C \to W'$ factors through U', then we get a commuting diagram as the solid part of

$$\begin{tikzcd}
T \arrow[rr] \arrow[dr, dashed, "k"'] \arrow[ddd] & & W \arrow[ddd] \\
& U \arrow[ur] \arrow[d] \arrow[dr, phantom, "\lrcorner" very near start] & \\
& U' \arrow[dr] & \\
C \arrow[rr] \arrow[ur] \arrow[uur, dashed, "l"] & & W',
\end{tikzcd}$$

hence a map k as indicated by Lemma 4.4.23, using monicity of s_U. By hypothesis, $U \to U'$ is in $J_{\mathsf{S}}^{\perp}$, so there is a unique map $l: C \to U$ making both triangles commute. By composing it with $U \to W$, we get the desired diagonal filler. □

This is the first crucial lemma of this section, which states that vertical composition in $\mathbb{D}^0(\mathbb{C})$ preserves $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$-cartesianness, and will be used to inductively prove that plays admit cartesian restrictions under the condition that seeds do (Theorem 4.4.30). Explicitly:

Lemma 4.4.27. *If any two vertically composable double cells of* $\mathsf{Cospan}(\widehat{\mathbb{C}})$ *are both in* $\mathbb{D}^0(\mathbb{C})$ *and* $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$*, and are cartesian in the latter, then their vertical composite is again cartesian (in the latter).*

Proof. Consider any two composable vertical morphisms $\langle U \rangle = (Y \to U \leftarrow X)$ and $\langle V \rangle = (Z \to V \leftarrow Y)$, and similarly $\langle U' \rangle$ and $\langle V' \rangle$, together with composable cartesian double cells $\alpha: \langle U \rangle \to \langle U' \rangle$ and $\beta: \langle V \rangle \to \langle V' \rangle$. To show that the composite is cartesian, it is enough by Lemma 4.4.22 to show that it has the shape of the front face of (4.10), i.e., that its top square is a pullback and that $U \bullet V \to U' \bullet V'$ is right-orthogonal to J_{S}.

Because $\langle U \rangle \to \langle U' \rangle$ is cartesian, by Lemma 4.4.22, the left face of

$$\begin{tikzcd}
Y \arrow[rr] \arrow[dr] \arrow[dd] \arrow[drr, phantom, "\lrcorner" very near start] \arrow[ddr, phantom, "\lrcorner" very near start] & & V \arrow[dr] \arrow[dd] & \\
& U \arrow[rr, crossing over] \arrow[ur, phantom, "\ulcorner" very near end] & & U \bullet V \arrow[dd] \\
Y' \arrow[rr] \arrow[dr] & & V' \arrow[dr] & \\
& U' \arrow[rr] \arrow[from=uu, crossing over] & & U' \bullet V',
\end{tikzcd}$$

is a pullback, so by two applications of Corollary 4.3.15 and Corollary 4.2.7 respectively, its front face is a 1D-pullback and its right one is a pullback. By Lemma 4.4.26, the obtained map $U \bullet V \to U' \bullet V'$ is thus in $J_{\mathsf{S}}^{\perp}$. Since $\langle V \rangle \to \langle V' \rangle$ is cartesian, by Lemma 4.4.22, the left-hand square below is a pullback, hence so is the right-hand one by the pullback lemma:

$$\begin{tikzcd}
Z \arrow[r] \arrow[d] & Z' \arrow[d] \\
V \arrow[r] & V'
\end{tikzcd}
\qquad
\begin{tikzcd}
Z \arrow[r] \arrow[d] & Z' \arrow[d] \\
U \bullet V \arrow[r] & U' \bullet V',
\end{tikzcd}$$

which concludes the proof. □

We will also need to start the induction somewhere, so we need to show that moves admit cartesian restrictions when seeds do (Lemma 4.4.29), which is based on the following lemma:

Lemma 4.4.28. *For any pushout*

$$\begin{tikzcd}
id^\bullet_Z \arrow[r, "id^\bullet_h"] \arrow[d, "k"'] & id^\bullet_{Z'} \arrow[d] \\
P \arrow[r] & P' \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{4.12}$$

in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$ *where* $h \in \mathbb{D}^0(\mathbb{C})_h(Z, Z')$ *and* $k \in \mathbb{D}(\mathsf{S})_H(id^\bullet_Z, P)$*,* P' *is a play and* $P \to P'$ *is cartesian and lies in* $\mathbb{D}^0(\mathbb{C})_H$ *(hence also in* $\mathbb{D}(\mathsf{S})_H$*).*

Proof. The fact that $P \to P'$ lies in $\mathbb{D}^0(\mathbb{C})_H$ follows form 1D-injectivity of h, stability of monos under pushout in Set, and the fact that pushouts along monos are pullbacks by adhesivity of Set. Let us show the other properties by induction on the play.

We prepare the induction by treating the case of moves. Let us thus assume that $P = \langle M \rangle$ is a move. We know that any move $\langle M \rangle$ is a pushout of some seed $\langle M_0 \rangle$. By Lemma 4.4.19, we may assume that it is the pushout of $\langle M_0 \rangle$ along a morphism $Z_0 \to Z'_0$, where Z_0 is the cofree invariant position of $\langle M_0 \rangle$. Since Z_0 is the cofree invariant position of $\langle M_0 \rangle$, by Lemma 4.4.20, we know that Z'_0 is the cofree invariant position of $\langle M \rangle$. Therefore, $id^\bullet_Z \to \langle M \rangle$ factors as $id^\bullet_Z \to id^\bullet_{Z'_0} \to \langle M \rangle$. Now, we define Z'' as the pushout below and $id^\bullet_{Z''} \to \langle P' \rangle$ by its universal property:

$$\begin{tikzcd}
id^\bullet_Z \arrow[r] \arrow[d] & id^\bullet_{Z'} \arrow[d] \arrow[ddr, bend left] & \\
id^\bullet_{Z'_0} \arrow[r] \arrow[dr] & id^\bullet_{Z''} \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dashed] & \\
& \langle M \rangle \arrow[r] & \langle P' \rangle .
\end{tikzcd}$$

Now, two applications of the pushout lemma give that

$$\begin{tikzcd}
id^\bullet_{Z_0} \arrow[r] \arrow[d] & id^\bullet_{Z''} \arrow[d] \\
\langle M_0 \rangle \arrow[r] & \langle P' \rangle
\end{tikzcd}$$

is a pushout, and since $Z_0 \to Z''$ is 1D-injective (as the composite of two 1D-injective maps), $\langle P' \rangle$ is a move, which we also call $\langle M' \rangle$.

Let us now show that the obtained morphism $\langle M \rangle \to \langle M' \rangle$ is cartesian using Lemma 4.4.22, i.e., by showing that it has the shape of the front face of (4.10). By the pushout lemma, the top square

$$\begin{tikzcd}
Y \arrow[r] \arrow[d] & Y' \arrow[d] \\
M \arrow[r] & M'
\end{tikzcd}$$

is a pushout along $Y \to M$, which is monic, so it is a pullback by adhesivity (Lemma 4.2.4). Moreover, to show that $M \to M'$ is right-orthogonal to J_{S}, we take any $T \to C$ in J_{S} and commuting square as the solid part of

$$\begin{tikzcd}
T \arrow[r] \arrow[d] & M \arrow[d] \\
C \arrow[r] \arrow[ur, dashed] & M'.
\end{tikzcd}$$

Since $M \to M'$ is an isomorphism in dimensions > 1 and C is a representable of dimension > 1, $C \to M'$ can be factored uniquely as shown above. Now, since T is tight and $M \to M'$ is 1D-injective, by Lemma 4.4.16, we get that the top-left triangle commutes as well, hence $C \to M$ is the desired diagonal filler.

Now that we have shown that moves are stable under pushouts of the desired form and that the resulting morphism is cartesian, we proceed to show that it is also the case for arbitrary plays by induction on $\langle P \rangle$.

If $Y \to P \leftarrow X$ contains zero moves, then the result is obvious. If $Y \to P \leftarrow X$ contains at least one move, we decompose it as $\langle M \rangle \bullet \langle U \rangle$, for some move $T \to M \leftarrow X$ and play $Y \to U \leftarrow T$ containing fewer moves than P.

Because the square below

$$\begin{tikzcd}
Z \arrow[dr, dashed] \arrow[drr, bend left] \arrow[ddr, bend right] & & \\
& T \arrow[r] \arrow[d] & U \arrow[d] \\
& M \arrow[r] & P \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

is a pushout along a monomorphism, hence a pullback, we obtain a unique dashed map as shown such that the diagram commutes. Thus, $id^{\bullet}_Z \to P$ factors as a vertical composite of two cells $id^{\bullet}_Z \to U$ and $id^{\bullet}_Z \to M$. By Lemma 4.4.21, the desired pushout (4.12) is the vertical composite of the following two pushouts:

$$\begin{tikzcd}
id^{\bullet}_Z \arrow[r] \arrow[d] & id^{\bullet}_{Z'} \arrow[d] \\
\langle U \rangle \arrow[r] & \langle U' \rangle \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}
\qquad
\begin{tikzcd}
id^{\bullet}_Z \arrow[r] \arrow[d] & id^{\bullet}_{Z'} \arrow[d] \\
\langle M \rangle \arrow[r] & \langle M' \rangle. \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

Since $\langle U' \rangle$ and $\langle M' \rangle$ are plays by induction hypothesis, so is $\langle P' \rangle$. Moreover, by induction hypothesis again, $\langle U \rangle \to \langle U' \rangle$ and $\langle M \rangle \to \langle M' \rangle$ are cartesian, and therefore, so is $\langle P \rangle \to \langle P' \rangle$, by Lemma 4.4.27. □

Remark. *The morphism $P \to P'$ thus computed is not opcartesian in general. To see this, consider the signature S_π of the π-calculus, and push the play $P = \langle o_{m,c,d} \rangle$ along $[m] \to [n]\, {}_a|_c\, [m]$ to get P'. Now, P clearly has a morphism to $\sigma_{n,a,m,c,d}$ (given by $\mathsf{S}_\pi(\sigma)$), but P' does not, because its final position contains a player of type $[n]$, while the final position of $\sigma_{n,a,m,c,d}$ only contains two players of type $[n+1]$ and $[m]$ respectively.*

Lemma 4.4.29. *If seeds admit cartesian restrictions in $\mathbb{D}(\mathsf{S})$, then so do moves.*

Proof. Consider any move as in (4.8) and horizontal morphism $h\colon X' \to X$. We start by forming the cube

$$\begin{tikzcd}
Z'_0 \arrow[rr] \arrow[dr, dashed] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & Z' \arrow[dr] \arrow[dd] & \\
& X'_0 \arrow[rr] \arrow[dd, "h_0"', crossing over] \arrow[dr, phantom, "\lrcorner", very near start] & & X' \arrow[dd, "h"] \arrow[ul, phantom, "\ulcorner", very near start] \\
Z_0 \arrow[rr] \arrow[dr] & & Z \arrow[dr] & \\
& X_0 \arrow[rr] & & X, \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd} \tag{4.13}$$

where the dashed arrow is obtained by universal property of pullback. By the pullback lemma, the left-hand face is again a pullback. Now, by Lemma 4.4.19, we may assume that $Z_0 \to X_0$ is monic, so by adhesivity the top face of (4.13) is again a pushout.

By Lemma 4.4.28, the morphism $\langle\mu\rangle \to \langle M\rangle$ is cartesian. By hypothesis, we obtain a cartesian lifting of $\langle\mu\rangle$ along h_0, say $Y'_0 \to U'_0 \leftarrow X'_0$. We get a morphism $Z'_0 \to U'_0$ by composing $Z'_0 \to X'_0$ and $X'_0 \to U'_0$, and then a morphism $Z'_0 \to Y'_0$ by universal property of pullback, remembering that, by Lemma 4.4.22, the square

$$\begin{tikzcd}
Y'_0 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & Y_0 \arrow[d] \\
U'_0 \arrow[r] & \mu
\end{tikzcd}$$

is a pullback. We may thus push the obtained lifting along $id^{\bullet}_{Z'_0} \to id^{\bullet}_{Z'}$ to obtain a play $\langle U'\rangle$ and a cartesian morphism $\langle U'_0\rangle \to \langle U'\rangle$ by Lemma 4.4.28. It also induces, by universal property of pushout, a morphism $\langle U'\rangle \to \langle M\rangle$ in $\mathsf{Cospan}(\widehat{\mathbb{C}})_H$ as in

$$\begin{tikzcd}
id^{\bullet}_{Z'_0} \arrow[rr] \arrow[dr] \arrow[dd] & & id^{\bullet}_{Z'} \arrow[dr] \arrow[dd] & \\
& \langle U'_0\rangle \arrow[rr] \arrow[dd, crossing over] & & \langle U'\rangle \arrow[dd, dashed] \arrow[ul, phantom, "\ulcorner", very near start] \\
id^{\bullet}_{Z_0} \arrow[rr] \arrow[dr] & & id^{\bullet}_{Z} \arrow[dr] & \\
& \langle\mu\rangle \arrow[rr] & & \langle M\rangle. \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

We want to show that $\langle U'\rangle$ is the cartesian restriction of $\langle M\rangle$ along h. In order to do that, we first need to show that $\langle U'\rangle \to \langle M\rangle$ is a morphism of plays, i.e., that it belongs to $\mathbb{D}(\mathsf{S})_H$.

Now consider the following cubes:

$$\begin{tikzcd}
X_0' \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & U_0' \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& X' \arrow[rr] \arrow[dd] & & U' \arrow[dd] \\
X_0 \arrow[rr] \arrow[dr] & & \mu \arrow[dr] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& X \arrow[rr] & & M
\end{tikzcd}
\qquad
\begin{tikzcd}
Z_0' \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & Z' \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& Y_0' \arrow[rr] \arrow[dd] & & Y' \arrow[dd] \\
Z_0 \arrow[rr] \arrow[dr] & & Z \arrow[dr] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& Y_0 \arrow[rr] & & Y,
\end{tikzcd}$$

where, in the left-hand case, both pushouts are obtained by the pushout lemma. In the left-hand cube, by pointwise adhesivity and Corollary 4.3.15, we obtain that $U' \to M$ is 1D-injective and that the front face is a 1D-pullback. In the right-hand cube, Corollary 4.3.15 entails that $Y' \to Y$ is 1D-injective. This entails in particular that $\langle U' \rangle \to \langle M \rangle$ indeed is a morphism of plays.

It remains to show that $\langle U' \rangle \to \langle M \rangle$ is cartesian, for which by Lemma 4.4.22 it is sufficient to show that it has the shape of the front face of (4.10).

First, the upper square is a pullback by pointwise application of Lemma 4.2.6 in

$$\begin{tikzcd}
Y_0' \arrow[rr] \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & & U_0' \arrow[dr] \arrow[dd] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& Y' \arrow[rr] \arrow[dd] & & U' \arrow[dd] \\
Y_0 \arrow[rr] \arrow[dr] & & \mu \arrow[dr] \arrow[dr, phantom, "\ulcorner", very near end] & \\
& Y \arrow[rr] & & M.
\end{tikzcd}$$

So the only point left to show is that $U' \to M$ lies in $J_{\mathsf{S}}^{\perp}$. To show this, we consider any morphism $T \to C$ in J_{S} and commuting square

$$\begin{tikzcd}
T \arrow[r] \arrow[d] & U' \arrow[d] \\
C \arrow[r] & M
\end{tikzcd}$$

and show that there is a unique diagonal filler. Uniqueness follows from the fact that $U' \to M$ is 1D-injective and that $C \cong \Sigma_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(C))$, so we only need to show that there exists such a diagonal filler.

First, since $\mu \to M$ is an isomorphism in dimensions > 1 and C is a representable of dimension > 1, we know that $C \to M$ factors through $\mu \to M$ in a unique way. We now want to show that $T \to U'$ factors through $U_0' \to U'$ in such a way that

$$\begin{tikzcd}
T \arrow[r, dashed] \arrow[d] \arrow[rr, bend left] & U'_0 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & U' \arrow[d] \\
C \arrow[r] \arrow[rr, bend right] & \mu \arrow[r] & M
\end{tikzcd}$$

commutes.

Stepping back a little, let us recall a cube considered above, as below left

$$\begin{tikzcd}
T \arrow[dr, dashed] \arrow[ddd] \arrow[ddddr] \arrow[rrrrdd, bend left] & & & & \\
& X'_0 \arrow[rr] \arrow[dr] \arrow[ddd] \arrow[dr, phantom, "\lrcorner", very near start] & & U'_0 \arrow[dr] \arrow[ddd] & \\
& & X' \arrow[rr, hook] \arrow[ddd] \arrow[dr, phantom, "\lrcorner", very near start] & & U' \arrow[ddd] \\
C \arrow[drrr, bend left=10] & & & & \\
& X_0 \arrow[rr] \arrow[dr] & & \mu \arrow[dr] & \\
& & X \arrow[rr] & & M
\end{tikzcd}
\qquad
\begin{tikzcd}
T \arrow[r] \arrow[d] & U'_0 \arrow[d] \\
C \arrow[r] \arrow[ur, dashed] & \mu,
\end{tikzcd}$$

where we added the map $T \to X_0$ given by $\mathsf{S}(C \to \mu)$. By Lemma 4.4.24 on the front face, using tightness of T, we obtain a unique arrow $T \to X'$, which further gives by universal property of pullback a unique dashed arrow $T \to X'_0$ making everything commute. In particular, we obtain a square as the solid part above right by composing the arrow $T \to X'_0$ we just obtained with $X'_0 \to U'_0$.

But now $U'_0 \to \mu$ is in $J_{\mathsf{S}}^{\perp}$, so there is a unique dashed diagonal map as shown making both triangles commute, which gives rise to a map $C \to U'$ by composition, hence the result. □

Finally, we state and prove our first fibredness criterion:

Theorem 4.4.30. *If seeds admit cartesian restrictions in* $\mathbb{D}(\mathsf{S})$*, then* $\mathbb{D}(\mathsf{S})$ *is fibred.*

Proof. Let us consider any play $Y \to P \leftarrow X$ and show that its cartesian restriction along $X' \to X$ in $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$ lies in $\mathbb{D}(\mathsf{S})$, which is enough by Lemma 4.4.3. We proceed by induction on $Y \to P \leftarrow X$. If it is the composite of 0 moves, then $X \to P$ and $Y \to P$ are isomorphisms and the result is obvious. If it is the composite of $n+1$ moves, then it can be decomposed as $\langle M \rangle \bullet \langle U \rangle$ for some move $T \to M \leftarrow X$ and play $Y \to U \leftarrow T$. By Lemma 4.4.29, we know that $\langle M \rangle$ admits a cartesian restriction along $X' \to X$, say $T' \to V' \leftarrow X'$. Furthermore, by induction hypothesis, $\langle U \rangle$ admits a cartesian restriction along $T' \to T$, say $Y' \to U' \leftarrow T'$. By Lemma 4.4.27, the vertical composition of $\langle V' \rangle \to \langle M \rangle$ and $\langle U' \rangle \to \langle U \rangle$ is cartesian, hence the result. □

4.4.4 Cartesian Lifting of Seeds

Required: 4.4.3.
Recommended: $\varnothing$.

In the previous section, we have shown that $\mathbb{D}(\mathsf{S})$ is fibred as soon as seeds admit cartesian restrictions in $\mathbb{D}(\mathsf{S})$. In this section, we exhibit sufficient conditions for the latter to be the case. I.e., possibly under additional hypotheses, in the setting of (4.10), if $\langle U \rangle$ is a seed, then its restriction $\langle U' \rangle$ is a play and (h, h', h'') is a morphism of plays.

The basic idea of our proof is that there are two possible cases: either X' "contains all of" X, or it does not. In more precise terms, either h is a retraction, or it is not. In both cases, for the given seed μ, we

- first construct a candidate restriction $\langle U' \rangle$,
- prove that it is indeed a play and that the morphism $\langle U' \rangle \to \langle \mu \rangle$ is a morphism of plays,
- and then finally show that it is a cartesian lifting of $\langle \mu \rangle$ along h by showing that it has the shape of the front face of (4.10).

The main difference between the two cases is that, in the first one, X can be thought of as a sub-position of X', so we basically extend μ so that it is played from all of X'. By contrast, in the second case, X' does not contain X, so it is impossible to play μ from it, and the restriction consists of all the "pieces" of μ that can be played from X'.

Let us make a first hypothesis that will be useful throughout the whole proof. It is equivalent to asking that moves never erase channels, in the sense that, if a channel occurs in the initial position of a move, then it also does in its final position. The hypothesis is the following:

Definition 4.4.31. *A signature* S *is* persistent *when for any seed* $Y \to M \leftarrow X$*, the morphism* $Z \to X$ *from its cofree invariant position is an isomorphism in dimension* 0*.*

Lemma 4.4.32. *The seeds of any persistent signature admit cartesian liftings along retractions.*

Proof. Consider any such signature S. By Lemma 4.4.2, it is enough to prove that the cartesian lifting in $\mathsf{Cospan}_{J_{\mathsf{S}}}(\widehat{\mathbb{C}})$ lies in $\mathbb{D}(\mathsf{S})$. Consider any seed $Y \to M \leftarrow X$ and 1D-injective retraction $h: X' \to X$. Since h is a retraction, there is a section $h': X \to X'$ such that $hh' = id_X$. We call Z the cofree invariant position of $Y \to M \leftarrow X$ and define Z' as the pullback

$$\begin{tikzcd}
Z \arrow[d, "u"'] \arrow[dr, dashed, "r'"] \arrow[drr, equal, bend left] & & \\
X \arrow[dr, "h'"'] & Z' \arrow[r, "r"] \arrow[d, "u'"'] \arrow[dr, phantom, "\lrcorner", very near start] & Z \arrow[d, "u"] \\
& X' \arrow[r, "h"'] & X
\end{tikzcd}$$

and $r' \colon Z \to Z'$ by its universal property. As a section, r' is mono, so in particular 1D-injective. We know that u is an isomorphism in dimension 0 by persistence, and so is u' as a pullback of u, which entails by Lemma 4.2.1 in the opposite category that the left-hand square above is a pushout in dimension 0. But in dimensions > 0, h is an isomorphism, hence so are h' and r' (as a pullback of h'), so by the same argument, the left-hand square is also a pushout in dimensions > 0. It is thus a pushout in all dimensions, hence a pushout in $\widehat{\mathbb{C}}$, since colimits are computed pointwise in presheaf categories.

We define the cospan $\langle M' \rangle$ as the pushout

$$\begin{tikzcd}
id^{\bullet}_{Z} \arrow[r, "id^{\bullet}_{r'}"] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & id^{\bullet}_{Z'} \arrow[d, "\beta"] \arrow[dr, "id^{\bullet}_{r}"] & \\
\langle M \rangle \arrow[r, "\alpha'"'] \arrow[drr, equal, bend right] & \langle M' \rangle \arrow[dr, dashed, "\alpha"'] & id^{\bullet}_{Z} \arrow[d] \\
 & & \langle M \rangle
\end{tikzcd} \tag{4.14}$$

and $\alpha \colon \langle M' \rangle \to \langle M \rangle$ by its universal property. Now, if we define $(l, k, \tilde{h}) = \alpha$, $(l', k', \tilde{h}') = \alpha'$, and $(l'', k'', h'') = \beta$, we can assume without loss of generality that $h'' = u'$ and that $\tilde{h}' = h'$, since both

$$\begin{tikzcd}
Z \arrow[r, "r'"] \arrow[d, "u"'] \arrow[dr, phantom, "\ulcorner", very near end] & Z' \arrow[d, "u'"] \\
X \arrow[r, "h'"'] & X'
\end{tikzcd} \quad \text{and} \quad \begin{tikzcd}
id^{\bullet}_{Z} \arrow[r, "id^{\bullet}_{r'}"] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & id^{\bullet}_{Z'} \arrow[d, "\beta"] \\
\langle M \rangle \arrow[r, "\alpha'"'] & \langle M' \rangle
\end{tikzcd}$$

are pushouts. Similarly, by the pushout lemma, we get that

$$\begin{tikzcd}
Z' \arrow[r, "r"] \arrow[d, "u'"'] \arrow[dr, phantom, "\ulcorner", very near end] & Z \arrow[d, "u"] \\
X' \arrow[r, "h"'] & X
\end{tikzcd} \quad \text{and} \quad \begin{tikzcd}
id^{\bullet}_{Z'} \arrow[r, "id^{\bullet}_{r}"] \arrow[d, "\beta"'] \arrow[dr, phantom, "\ulcorner", very near end] & id^{\bullet}_{Z} \arrow[d] \\
\langle M' \rangle \arrow[r, "\alpha"'] & \langle M \rangle
\end{tikzcd}$$

are pushouts, so we can assume that $\tilde{h} = h$. By Lemma 4.4.28 in the left-hand square of (4.14), we get that $\langle M' \rangle$ is a play. Now, by Lemma 4.4.28 in the right-hand square, we get that α is cartesian, so $\langle M' \rangle$ is a lifting of $\langle M \rangle$ along h. □

Remark. *The lemma above shows that, if a signature is persistent, then it admits cartesian liftings along retractions. The converse actually holds. Indeed, consider any signature* S *that is not persistent. This means that there is some move* m_0 *in the base category* $\mathbb{C}$ *whose seed* $\langle M \rangle$ *is such that the morphism from the cofree invariant position* Z *to* X *is not an isomorphism in dimension* 0*. Since that morphism is a monomorphism as the pullback of* $Y \to M$*, which is mono, it cannot be an epimorphism in dimension* 0 *(because all epi monos are isos in* Set*). This implies that there is a channel* $c \colon \star \to M$ *for some object* $\star$ *of dimension* 0 *that is in the image of* $X \xrightarrow{t} M$*, but not in that of* $Y \xrightarrow{s} M$*.*

Let us consider the move $\langle M \rangle$ *and try to build its cartesian lifting along* $\star + X \xrightarrow{[c,X]} X$*. If such a cartesian lifting* $\langle U' \rangle \to \langle M \rangle$ *exists, because it is cartesian,* $id_{\langle M \rangle}$ *must factor through it in a unique way, as in*

$$\begin{tikzcd}
Y \arrow[d] \arrow[rrd, bend left, equal] \arrow[rd, dashed] & & \\
M \arrow[rrd, bend left, equal] \arrow[rd, dashed] & Y' \arrow[r] \arrow[d] & Y \arrow[d] \\
X \arrow[u] \arrow[rrd, bend left, equal] \arrow[rd] & U' \arrow[r] & M \\
 & * + X \arrow[r] \arrow[u] & X \arrow[u].
\end{tikzcd}$$

Therefore, $U' \to M$ must be a retraction, but it is also 1D-injective so it is bijective in dimensions > 0. In particular, $U'(m)$ is isomorphic to $M(m)$ for all moves m. Now, since U' must be isomorphic to m_0 in dimensions > 0, it is necessarily the composite of a single m_0 move. But the only way to play m_0 from $* + X$ is to be isomorphic to $* + Y \xrightarrow{*+s} * + M \xleftarrow{*+t} * + X$. So we want to find a pair of morphisms as the dashed arrows in:

$$\begin{tikzcd}
* + Y \arrow[r, "{[g,g']}"] \arrow[d, "*+s"'] & Y \arrow[d, "s"] \\
* + M \arrow[r, "{[f,f']}"] & M \\
* + X \arrow[r, "{[c,X]}"'] \arrow[u, "*+t"] & X. \arrow[u, "t"']
\end{tikzcd}$$

In particular, it must be that $tc = f = sg$. But we chose c such that tc is not in the image of s, so this is impossible, hence $\langle M \rangle$ has no cartesian lifting along $[c, X]$.

When h is not a retraction, we need some more hypotheses to construct restrictions (and ensure that they are indeed cartesian). We will need three hypotheses. The first one, *monolithicity*, restricts the possible ways to model interactions between players. It is not necessary, but makes proofs much simpler. The second one, *fragmentation*, forces all interactions to have restrictions to single players. This condition is necessary for seeds to admit cartesian restrictions (otherwise, the seed corresponding to an interaction could be restricted along one of the players). The last condition, *separation*, states that, if a player is created by an interaction, then it is the avatar of a single one of the players involved in that interaction. This is related to the notion of view: to define the vie of the created player, we take the view of the player it was created by. This condition is also necessary, otherwise a player created during an interaction could end up being created multiple times in a restriction, which would break 1D-injectivity.

First, we want to limit the possible interactions between moves:

Definition 4.4.33. *A signature is* monolithic *when for all morphisms of plays $\alpha\colon (Y \to M \leftarrow X) \to (Y' \to M' \leftarrow X')$ between any two seeds, if X is not a representable, then $M = M'$.*

A signature is monolithic roughly when restricting a move $Y' \to M' \leftarrow X'$ describing an interaction to a strict sub-position X of X' cannot yield an interaction. In other words, interactions cannot be broken down into smaller interactions (e.g., an interaction between three players cannot be broken down into an interaction between only two of them). Though it is possible to relax this limitation, this would not only induce an explosion of the number of cases to analyse, but we would also need to put another kind of restriction, which would more difficult to check, which is why we decided to stick to a simple case. Moreover, such "partial interactions" do not appear to bring much expressiveness to our framework.

Remark. *Since the base category is direct, monolithicity ensures that, if there is a morphism α between seeds as above, and X is not a representable, then $\alpha = (id_Y, id_M, id_X)$.*

A second hypothesis that we make says that there should exist a "biggest part" of any move from the point of view of any player involved in it. Basically, we want to ensure that any seed $Y \to \mu \leftarrow X$ has restrictions along all morphisms of positions $h: X' \to X$, and we prove this property by pasting together the "biggest part" of what each player in X' sees of μ when restricted along h. We here give a notion that entails the desired property, and basically amounts to asking that seeds admit cartesian restrictions along morphisms of the form $\mathsf{y}_d \to X$ with respect to other seeds (as opposed to arbitrary plays):

Definition 4.4.34. *A signature is* fragmented *if and only if for all seeds $Y \xrightarrow{s} M \xleftarrow{t} X$ and players $x: d \to X$, there exists a seed $Y_{M,x} \xrightarrow{s_{M,x}} M_{|x} \xleftarrow{t_{M,x}} d$ and a morphism $f_{M,x}: M_{|x} \to M$ in $\mathbb{C}$ such that:*

(a) *the top square of $\mathsf{S}(f_{M,x})$ is a pullback, and*

(b) *for any seed $Y'' \to M'' \leftarrow X''$ and commuting diagram as the solid part below, there is a map $M'' \to M_{|x}$ making the diagram commute.*

$$\begin{tikzcd}
M'' \arrow[drr] \arrow[dr, dashed] & & \\
X'' \arrow[u] \arrow[dr] \arrow[drr] & M_{|x} \arrow[r, "f_{M,x}"'] & M \\
& d \arrow[u] \arrow[r] & X \arrow[u]
\end{tikzcd} \tag{4.15}$$

Remark. *Since $f_{M,x}$ is 1D-injective and M'' is a representable of dimension > 1, the morphism $M'' \to M_{|x}$ in the hypothesis above is necessarily unique.*

Lemma 4.4.35. *If a signature S is persistent, monolithic, and fragmented, then each morphism $\mathsf{S}(f_{M,x})$ is a cartesian lifting of $\langle M \rangle$ along x in $\mathbb{D}(\mathsf{S})$.*

Proof. By Lemma 4.4.22, it is enough to prove that the top square of $\mathsf{S}(f_{M,x})$ is a pullback and that $f_{M,x}$ in $J_{\mathsf{S}}^{\perp}$. The first point holds by *(a)*. To prove the second one, consider any morphism $T \to C$ in J_{S} and commuting square

$$\begin{tikzcd}
T \arrow[r] \arrow[d] & M_{|x} \arrow[d, "f_{M,x}"] \\
C \arrow[r] & M.
\end{tikzcd}$$

We need to show that there is a unique diagonal filler $C \to M_{|x}$. By Corollary 4.4.25, we obtain a unique morphism $T \to d$ making the diagram below left commute:

$$\begin{tikzcd}
T \arrow[r, dashed] \arrow[d] \arrow[dr] & d \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\
C \arrow[rr, bend right] & M_{|x} \arrow[r] & M
\end{tikzcd}
\qquad
\begin{tikzcd}
C \arrow[drr] & & \\
T \arrow[u] \arrow[dr] \arrow[drr] & M_{|x} \arrow[r] & M \\
& d \arrow[u] \arrow[r] & X, \arrow[u]
\end{tikzcd}$$

which may be arranged as on the right to have the shape of (4.15). We thus conclude by *(b)*. □

Lemma 4.4.35 exhibits cartesian liftings along players $d \to X$. Let us now consider more general cases, assuming a third property saying that each player involved in a *synchronisation* M (i.e., a seed whose initial position contains several players) is related to at most one of the "biggest parts" of M, in the sense of our above explanation of fragmentedness.

Definition 4.4.36. *A signature* S *is* separated *if it is fragmented and, for all moves* $\mu \in \mathrm{ob}(\mathbb{C}_{|\geq 2})$ *with seed* $\mathsf{S}(\mu) = (Y \to \mu \leftarrow X)$, *players* $d \in \mathrm{ob}(\mathbb{C}_{|1})$, $x_1{:}\, d_1 \to X$ *and* $x_2{:}\, d_2 \to X$, *and* $y_1{:}\, d \to \mu_{|x_1}$ *and* $y_2{:}\, d \to \mu_{|x_2}$ *making*

$$\begin{tikzcd}
d \arrow[r, "y_1"] \arrow[d, "y_2"'] & \mu_{|x_1} \arrow[d, "f_{\mu,x_1}"] \\
\mu_{|x_2} \arrow[r, "f_{\mu,x_2}"'] & \mu
\end{tikzcd}$$

commute, we have $x_1 = x_2$ *(and hence* $f_{\mu,x_1} = f_{\mu,x_2}$*).*

Remark. *Separation really only says something in the case where the diagonal* $x{:}\, d \to \mu$ *does not factor through* X*. Indeed, if it does, then by the properties of 1D-pullbacks,* x *also factors through* x_1 *and* x_2*, hence by directedness of* $\mathbb{C}$ *is in fact equal to* x_1 *and* x_2*.*

Remark. *Separation is related to the notion of* view *in game semantics (and indeed, in the cases we are interested in, separation is derived from what is called the axiom of views in [50]). It basically states that any player that is created in a move* M *is created by at most one player.*

When h is not a retraction, the restriction of a seed along h has a particular form that we call a "quasi-move", which basically consists of several moves played "in parallel", i.e., independently from one another.

Definition 4.4.37. *A* quasi-move *is any cospan that is isomorphic to a pushout of the form*

$$\begin{tikzcd}
\sum_{i \in n} id^{\bullet}_{Z_i} \arrow[r, "id^{\bullet}_h"] \arrow[d] & id^{\bullet}_Z \arrow[d] \\
\sum_{i \in n} \langle M_i \rangle \arrow[r] & \langle U \rangle, \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

in $\mathbb{D}^0(\mathbb{C})_H$*, where the* $\langle M_i \rangle$*'s are seeds.*

Lemma 4.4.38. *Every quasi-move is a play.*

Proof. Let $\langle M_i \rangle = (Y_i \xrightarrow{s_i} M_i \xleftarrow{t_i} X_i)$ for all $i \in n$. By Lemma 4.4.28, it is enough to show that $\sum_i \langle M_i \rangle$ is a play. We proceed by induction on n. If $n = 0$, the result is trivial. Otherwise, we have

$$\sum_i \langle M_i \rangle \cong (\langle M_1 \rangle + \sum_{i>1} id^{\bullet}_{X_i}) \bullet (id^{\bullet}_{Y_1} + \sum_{i>1} \langle M_i \rangle).$$

By induction hypothesis, both components are plays, so by Lemma 4.4.21, the composite also is. □

We can now exhibit the construction of the restriction of a seed $Y \xrightarrow{s} M \xleftarrow{t} X$ along a morphism $h : X' \to X$ that is not a retraction.

Theorem 4.4.39. *The seeds of any persistent, monolithic, and separated signature* S *admit cartesian liftings in* $\mathbb{D}(\mathsf{S})$*.*

Proof. By Lemma 4.4.32, it is enough to deal with the case where h is not a retraction. We first build the candidate restriction. Let Z be the cofree invariant position of $\langle M \rangle$. By fragmentedness, we know that for each player $(d, x) \in \mathrm{pl}(X)$, there exists a seed $Y_{M,x} \xrightarrow{s_{M,x}} M_{|x} \xleftarrow{t_{M,x}} d$ and a morphism $(l_{M,x}, f_{M,x}, x) : \langle M_{|x} \rangle \to \langle M \rangle$ satisfying *(a)* and *(b)* of Definition 4.4.34. Letting $Z' = X' \times_X Z$, for any player $(d, x) \in \mathrm{pl}(X')$ we may thus construct by universal property of pullback a map $r_{M,x}$ as in

$$\begin{tikzcd}
 & & Z_{M,hx} \arrow[rr] \arrow[dr] \arrow[dr, phantom, very near start, "\lrcorner"] \arrow[dd] \arrow[ddll, dashed, bend right, "r_{M,x}"'] & & Y_{M,hx} \arrow[dr] \arrow[dd] \arrow[dd, phantom, very near start, "\lrcorner"] & \\
 & & & d \arrow[rr] \arrow[dd] \arrow[ddll, bend right, "x"'] & & M_{|hx} \arrow[dd] \\
Z' \arrow[rr, "r"] \arrow[dr] \arrow[dr, phantom, very near start, "\lrcorner"] & & Z \arrow[rr] \arrow[dr] \arrow[dr, phantom, very near start, "\lrcorner"] & & Y \arrow[dr] & \\
 & X' \arrow[rr, "h"'] & & X \arrow[rr] & & M,
\end{tikzcd} \tag{4.16}$$

where $Z_{M,hx} \to Z$ comes from the universal property of Z. We first want to show that

$$\begin{tikzcd}
\sum_{(d,x)\in\mathrm{pl}(X')} Z_{M,hx} \arrow[r, "{[r_{M,x}]_{(d,x)\in\mathrm{pl}(X')}}"] \arrow[d] & Z' \arrow[d] \\
\sum_{(d,x)\in\mathrm{pl}(X')} d \arrow[r, "{[x]_{(d,x)\in\mathrm{pl}(X')}}"'] & X'
\end{tikzcd} \tag{4.17}$$

is a pushout and that all involved maps are 1D-injective. First, it is a pullback: by Lemma 4.2.2, it suffices to show $Z_{M,hx} = d \times_{X'} Z'$ for each x, which follows by three consecutive applications of the pullback lemma in (4.16). Because it is a pullback of two 1D-injective maps, all of its maps are in fact 1D-injective. But then, recalling that the pullback of any isomorphism is in fact a pushout square, we have:

- in dimension 1, $\sum_{(d,x)\in\mathrm{pl}(X')} d \to X'$ is an isomorphism and

- in dimension 0, $Z' \to X'$ is an isomorphism by persistence,

so the square is a pushout in all dimensions, hence a proper pushout.

We now define our candidate restriction $\langle U' \rangle$ as the quasi-move below, and $\langle U' \rangle \to \langle M \rangle$ by its universal property:

$$\begin{tikzcd}
\sum_{(d,x)\in \mathrm{pl}(X')} id^{\bullet}_{Z'_{M,hx}} \arrow[r, "id^{\bullet}_{[r_{M,x}]_{(d,x)\in \mathrm{pl}(X')}}"] \arrow[d] & id^{\bullet}_{Z'} \arrow[d] \arrow[dr, "id^{\bullet}_{r}"] & \\
\sum_{(d,x)\in \mathrm{pl}(X')} \langle M_{|hx} \rangle \arrow[r, "{(l',k',h')}"'] \arrow[drr, bend right, "{[\mathsf{S}(f_{M,hx})]_{(d,x)\in \mathrm{pl}(X')}}"'] & \langle U' \rangle \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dashed, "{(l,k,\tilde{h})}"'] & id^{\bullet}_{Z} \arrow[d] \\
 & & \langle M \rangle .
\end{tikzcd}$$

First, $\langle U' \rangle$ is a quasi-move, and therefore a play by Lemma 4.4.38. Moreover, because (4.17) is a pushout, we can assume without loss of generality that it is the bottom square of the pushout defining $\langle U' \rangle$. Thus, $\tilde{h} = h$.

Furthermore, the morphism $[\mathsf{S}(f_{M,hx})]_{(d,x)\in \mathrm{pl}(X')}$ is in $\mathbb{D}(\mathsf{S})_H$. Indeed, that its bottom square is a 1D-pullback follows from Lemma 4.2.2; 1D-injectivity of its bottom component is 1D-injectivity of $h \circ [x]_{(d,x)\in \mathrm{pl}(X')}$; 1D-injectivity of its top component follows from that of its middle component. So let us consider an object c of dimension > 0 and two morphisms $f_1, f_2 \colon c \to \sum_{(d,x)\in \mathrm{pl}(X')} M_{|hx}$ that are equal when composed with $[f_{M,hx}]_{(d,x)\in \mathrm{pl}(X')}$. Since c is a representable, f_1 and f_2 must factor through one of the coproduct injections, so $f_i = inj_{(d_i,x_i)} \circ f'_i$ for some $f'_1 \colon c \to M_{|hx_1}$ and $f'_2 \colon c \to M_{|hx_2}$. We thus have a commuting diagram

$$\begin{tikzcd}
c \arrow[r, "f'_2"] \arrow[d, "f'_1"'] & M_{|hx_2} \arrow[d, "f_{M,hx_2}"] \\
M_{|hx_1} \arrow[r, "f_{M,hx_1}"'] & M.
\end{tikzcd}$$

We want to exhibit a square as above but with an object d of dimension 1 instead of c. If c is of dimension 1, we take $d = c$. Otherwise , by monolithicity, we know that $\mathsf{S}(c)$ must have a representable as its initial position, and since all initial positions must be tight, it must be a representable of dimension 1. We thus get a morphism $x \colon d \to c$ for some object d of dimension 1. We can now compose the diagram above with that morphism, which gives us a diagram on which we may apply the property of separation, which gives us that $hx_1 = hx_2$. Now,

$$\begin{aligned}
f_{M,hx_1} f'_1 &= [f_{M,hx}]_{(d,x)\in \mathrm{pl}(X')} \, inj_{(d_1,x_1)} f'_1 \\
&= [f_{M,hx}]_{(d,x)\in \mathrm{pl}(X')} f_1 \\
&= [f_{M,hx}]_{(d,x)\in \mathrm{pl}(X')} f_2 \\
&= [f_{M,hx}]_{(d,x)\in \mathrm{pl}(X')} \, inj_{(d_2,x_2)} f'_2 \\
&= f_{M,hx_2} f'_2 \\
&= f_{M,hx_1} f'_2 .
\end{aligned}$$

Therefore, since f_{M,hx_1} is 1D-injective and $c \cong \sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(c))$, we have that $f'_1 = f'_2$ by Proposition 4.4.13, hence $f_1 = f_2$, so $[f_{M,x}]_{(d,x)\in \mathrm{pl}(X')}$ is indeed 1D-injective.

We now want to show that the morphism $(l,k,h)\colon \langle U'\rangle \to \langle M\rangle$ is in $\mathbb{D}(\mathsf{S})_H$, i.e., that all its components are 1D-injective. We know that $[r_{M,x}]_{(d,x)\in\mathrm{pl}(X')}$ is bijective in dimensions > 0, as the pullback of $[x]_{(d,x)\in\mathrm{pl}(X')}$, which is by construction. Thus, the components of the map (l',k',h') are bijective in dimensions > 0, which entails that $\langle U'\rangle \to \langle M\rangle$ is in $\mathbb{D}(\mathsf{S})_H$.

Finally, let us prove that $\langle U'\rangle$ is the restriction of $\langle M\rangle$ along h. To this end, let us first prove that k is in $J_{\mathsf{S}}^{\perp}$: consider any $T \to C$ in J_{S} and commuting square

$$\begin{tikzcd}
T \arrow[r] \arrow[d] & U' \arrow[d] \\
C \arrow[r] & M.
\end{tikzcd}$$

By Corollary 4.4.25, we get a dashed map as in

$$\begin{tikzcd}
T \arrow[r, dashed] \arrow[d] \arrow[dr] & X' \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\
C \arrow[rr, bend right] & U' \arrow[r] & M,
\end{tikzcd}$$

which makes the diagram commute. If C were equal to M, then, since $\mathbb{C}$ is direct, $C \to M$ would be an identity, and so would $T \to X$ by Lemma 4.3.20, which contradicts the hypothesis that h is not a retraction. Thus, C is different from M. Therefore, by monolithicity, we know that T is representable, and of dimension 1 by tightness. Hence, $T \to X'$ factors as $T = d_0 \to \sum_{(d,x)\in\mathrm{pl}(X')} d \to X'$. We thus have the solid part of:

$$\begin{tikzcd}
T \arrow[r, equal] \arrow[d] & d_0 \arrow[r, "inj_{(d_0,x_0)}"] \arrow[d] & \sum_{(d,x)\in\mathrm{pl}(X')} d \arrow[r, "h'"] \arrow[d] & X' \arrow[r, "h"] \arrow[d] & X \arrow[d] \\
C \arrow[r, dashed] \arrow[rrrr, bend right] & M_{|hx_0} \arrow[r, "inj_{(d_0,x_0)}"] & \sum_{(d,x)\in\mathrm{pl}(X')} M_{|hx} \arrow[r, "k'"] & U' \arrow[r, "k"] & M,
\end{tikzcd} \tag{4.18}$$

which we can reorganise as the solid part of:

$$\begin{tikzcd}
C \arrow[dr, dashed] \arrow[drr] & & \\
T \arrow[u] \arrow[dr, equal] \arrow[drr] & M_{|hx_0} \arrow[r, "f_{M,hx_0}"'] & M \\
& d_0 \arrow[u] \arrow[r, "hx_0"'] & X \arrow[u]
\end{tikzcd}$$

on which we can use fragmentedness to get a dashed map making both diagrams commute. Since (4.18) commutes, it is obvious that $C \to U'$ obtained as the composite $C \to M_{|hx_0} \to U'$ is a desired diagonal filler. Uniqueness of $C \to U'$ is given by Proposition 4.4.13, since k is 1D-injective, and $C \cong \sum_{\mathsf{i}_1}(\Delta_{\mathsf{i}_1}(C))$.

Lastly, we prove that the right-hand square below is a pullback:

$$\begin{tikzcd}
Z' + \sum_{(d,x)\in\mathrm{pl}(X')} Y_{M,hx} \arrow[r] \arrow[d] & Y' \arrow[r] \arrow[d] & Y \arrow[d] \\
Z' + \sum_{(d,x)\in\mathrm{pl}(X')} M_{|hx} \arrow[r] & U' \arrow[r] & M.
\end{tikzcd} \tag{4.19}$$

By Lemma 4.2.2, the outer square is a pullback because the square below left (by Lemma 4.2.1) and each of the squares below right (by fragmentedness) are:

$$\begin{tikzcd}
Z' \arrow[r] \arrow[d, equal] & Y \arrow[d, hook] \\
Z' \arrow[r] & M
\end{tikzcd}
\qquad
\begin{tikzcd}
Y_{M,x} \arrow[r] \arrow[d] & Y \arrow[d] \\
M_{|x} \arrow[r] & M.
\end{tikzcd}$$

Moreover, the left-hand square in (4.19) is also a pullback by Lemmas 4.2.1, 4.4.28 (because we know that the top squares of cartesian liftings are pullbacks), and 4.2.2. Since $Z' + \sum_{(d,x)\in\mathrm{pl}(X')} M_{|x} \to U'$ is epi, Lemma 4.2.3 entails that the desired square is a pullback. □

Corollary 4.4.40. *For any persistent, monolithic, and separated signature* S*,* $\mathbb{D}(\mathsf{S})$ *is fibred.*

As an easy application, we get:

Proposition 4.4.41. $\mathbb{D}(\mathsf{S}_\pi)$ *is fibred.*

Proof. By Corollary 4.4.40, it suffices to verify that S_π is persistent, monolithic, and separated, which is routine. □

4.5 Perspectives

We have introduced a notion of signature for the sheaf-based approach to concurrent game semantics [50, 29]. We have shown how to, from such a signature, automatically build a pseudo double category of concurrent traces, which we interpret as plays in some sort of game. Finally, we have introduced the notion of fibredness, which allows us to build categories of plays from our pseudo double categories, and given two criteria to prove that a pseudo double category is fibred, one necessary and sufficient, the other simpler but only sufficient. The second criterion is still general enough to show that many interesting calculi yield fibred pseudo double categories, as illustrated by the example of the π-calculus, which can also be adapted to recover this property of a previous construction of CCS, as well as to show that the pseudo double category for HON games that we define in Chapter 5 is fibred.

The link between the pseudo double category we construct and games is further explored in Chapter 5, where we build a signature for HON games and investigate the link between the categories of plays we obtain and more traditional categories of plays.

Unfortunately, there are constructions that we would like to be able to do but that do not fit in this framework. One limitation of this framework is that moves are defined as pushouts along any (1D-injective) map, but for some calculi, we would like to be able to restrict the class of morphisms along which we are allowed to push. For example, in a π-calculus with match and mismatch operators, there would be a move $\varepsilon_{n,a,b}$ that may only be played when channels a and b are equal, and a move $\delta_{n,a,b}$ that may only be played when a and b are different. The first move is not difficult to define, we simply add morphisms $s, t\colon [n] \to \varepsilon_{n,a,b}$, quotient maps by $ss_i = ts_i$ and $ts_a = ts_b$, and define its seed to be $[n]_{a=b} \xrightarrow{s'} \varepsilon_{n,a,b} \xleftarrow{t'} [n]_{a=b}$, where $[n]_{a=b}$ is the representable $[n]$ whose channels a and b have been identified, or, more formally, the coequaliser of

$$\begin{tikzcd} 2 \cdot \star \arrow[r, shift left, "s_a"] \arrow[r, shift right, "s_b"'] & {[n]} \end{tikzcd}$$

and s' and t' are given by universal property of coequaliser applied to s and t respectively. This is not perfect however, in the sense that the obtained signature is not fragmented. The clean solution to overcome this problem is to define another move $\varepsilon^0_{n,a,b}$, with morphisms $s, t\colon [n] \to \varepsilon^0_{n,a,b}$ and $\partial\colon \varepsilon^0_{n,a,b} \to \varepsilon_{n,a,b}$ satisfying the obvious equations, and define its seed as $[n] \xrightarrow{s} \varepsilon^0_{n,a,b} \xleftarrow{t} [n]$. Finally, there would be a special kind of plays (called *closed-world plays*) that do not contain any move built from $\varepsilon^0_{n,a,b}$.

However, defining $\delta_{n,a,b}$ is trickier. Indeed, the only restriction to be able to play this move is that a and b should be different. So we should model it as an object $\delta_{n,a,b}$ with maps $s, t\colon [n] \to \delta_{n,a,b}$, with the morphisms in $\mathbb{C}$ quotiented by $ss_i = ts_i$, and define its seed to be $[n] \xrightarrow{s} \delta_{n,a,b} \xleftarrow{t} [n]$. However, since we are allowed to push along any morphism, nothing prevents us from building the move

$$\begin{tikzcd}
 & {[n]} \arrow[rr] \arrow[d] & & {[n]_{a=b}} \arrow[d, dashed] \\
 & \delta_{n,a,b} \arrow[rr] & & M \\
2 \cdot \star \arrow[uur] \arrow[ur] \arrow[rr] \arrow[dr, "{[s_a, s_b]}"'] & & \star \arrow[uur] \arrow[ur] \arrow[dr] & \\
 & {[n]} \arrow[uu] \arrow[rr] & & {[n]_{a=b},} \arrow[uu, dashed]
\end{tikzcd}$$

which is a move $\delta_{n,a,b}$ played by a player whose a and b channels are equal, exactly what we wanted to avoid! So, in this case, we would want to be able to push this move only along those morphisms that do not identify a and b. The only solution here is to allow only plays in which all moves m over $\delta_{n,a,b}$ are such that $m \cdot (ts_a) \neq m \cdot (ts_b)$ in closed-world plays. This is much more difficult in the sense that we are refusing plays based on the morphisms along which we push seeds, rather than simply forbidding some moves.

Another example where we do not want to be able to push along any morphism is that of HON games. Let us take the following position as a concrete example:

$$\xrightarrow{A} \overset{x}{\bullet} \xrightarrow{B} \overset{y}{\bullet} \xrightarrow{C} .$$

It consists of two players, x on the left and y on the right. The basic idea is that there are three kinds of moves in our calculus for HON games: Λ and @, which correspond to Opponent and Proponent moves, and β, which corresponds to the synchronisation of both moves. The Λ and @ moves should correspond to interacting with the environment, while the β moves should correspond to internal moves. Therefore, even though y is able to play @ on its own (by interacting with the environment), it should not be able to play it in the position drawn above, because such a move should be internal, and thus synchronised with a Λ move by x.

Moreover, the persistence property, which is necessary for our construction to admit restrictions along retractions, is sometimes too restrictive. Indeed, in the variant of HON games that we give in Chapter 5, the notion of *view* differs from the notion of view that was given in previous models of CCS and the π-calculus based on the same techniques [50, 29]. In order to recover the standard notion of view, one would need to add a new move to the base category, together with a seed that does not verify persistence.

Some possible future work on this framework would thus be to generalise it to accept some or all of the examples above.

Chapter 5

Justified Sequences in String Diagrams

5.1 Motivation

In this chapter, we study the link between two approaches to concurrent game semantics. At the level of plays, the first one is based on traditional notions of HON games, while the second one is based on *string diagrams*, as illustrated in Chapter 3. We reuse the machinery developed in Chapter 4 to define a pseudo double category of string diagrams for HON games and prove that it is fibred, thus allowing us to define categories of views and plays. We then set out to explore the link between the two approaches at the level of plays.

While the categories of plays we obtain are not the same, at the level of strategies, both approaches follow the same idea, by defining innocent strategies as sheaves over plays for a Grothendieck topology induced by the embedding of "views" into "plays". We make the link between the two approaches formal by proving that they in fact define equivalent categories of innocent strategies.

Two approaches to concurrent game semantics

Recent advances in concurrent game semantics have produced new game models for a non-deterministic, simply-typed λ-calculus on the one hand [97], and for CCS and the π-calculus on the other hand [49, 50, 29]. These models are based on categories of innocent and concurrent strategies that are defined in both cases as categories of *sheaves* over a site of plays.

The first model, by Tsukada and Ong, has proven to successfully extend the most fundamental results of game semantics (interpreting terms as innocent strategies and composing strategies to form a cartesian closed category of arenas and innocent strategies) to a non-deterministic λ-calculus. On the other hand, our approach aims to give a general framework to build game models for different calculi, in order to study translations between them.

There is a clear, yet informal relationship between the two approaches in that they both define innocent strategies as sheaves for a Grothendieck topology induced by embedding *views* into *plays*. However, despite this similarity, the notions of views and plays differ significantly. Indeed, Tsukada and Ong [97] define them as *justified sequences* of moves satisfying additional conditions, as in standard Hyland-Ong/Nickau game semantics [56, 87], while in [49, 50, 29], plays are defined as *ad hoc* string diagrams describing the game, close in spirit to ideas originally suggested by Melliès in a different setting (*circa* 2008, published as [83]). Since the notions of plays differ significantly, it is legitimate to wonder to what degree the approaches are related.

The level of views and plays

In this chapter, we go beyond this informal similarity and show a tight correspondence between the two approaches at the level of plays. Specifically, in Section 5.2, we design a pseudo double category $\mathbb{D}_{HON}$ of string diagrams for HON games whose vertical morphisms represent concurrent traces in HON games, just like we did for the π-calculus in Section 4.3. From $\mathbb{D}_{HON}$, we design, for each pair of arenas A and B, categories $\mathbb{E}(A \vdash B)$ and $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ respectively of plays and views for HON games.

In Tsukada and Ong's model, there are also standard categories $\mathbb{P}_{A,B}$ and $\mathbb{V}_{A,B}$ of plays and views as defined in standard game semantics (and which we call *TO-plays* and *TO-views* to disambiguate), with a notion of morphism inspired by Melliès's work [80]. In Sections 5.3 and 5.4, we show how to embed these categories into $\mathbb{E}(A \vdash B)$ and $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ respectively (whose objects we simply call *plays* and *views*).

We give two ways to define this embedding. The first way, which we explain in Section 5.3, goes through a third model, whose plays are proof trees in an *ad hoc* sequent calculus, and whose views are branches of those trees. The categories of trees and branches are equivalent to those of plays and views respectively, so they can be thought of as another possible representation of these objects. Trees may also be seen as a maximal parallelisation of TO-plays, while branches are simply equivalent to TO-views.

This definition is not very satisfactory, in the sense that trees are treated very informally, and a formal treatment (e.g., by defining proof trees as certain pointed presheaves over a well-chosen base category) would not make things easier than defining it directly. It however gives a good idea of how the embedding works, and what the discrepancy between our notion of play and that of Tsukada and Ong is, which is why we still choose to expose it. Then, in Section 5.4, we give a more formal definition of the embedding and show it verifies all the desired properties.

We thus obtain, for each pair of arenas A and B, a commuting square of embeddings of categories:

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r, hook, "\mathsf{i}_{TO}"] \arrow[d, "F^{\mathbb{V}}"'] & \mathbb{P}_{A,B} \arrow[d, "F"] \\
\mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[r, hook, "\mathsf{i}"'] & \mathbb{E}(A \vdash B),
\end{tikzcd} \tag{5.1}$$

where i_{TO} denotes the embedding of TO-views into TO-plays, i denotes the embedding of views into plays, and $F^{\mathbb{V}}$ and F denote the constructed embeddings, respectively from TO-views into views and from TO-plays into plays. Our first result is that all these embeddings are full and that $F^{\mathbb{V}}$ is an equivalence of categories (Theorem 5.3.11).

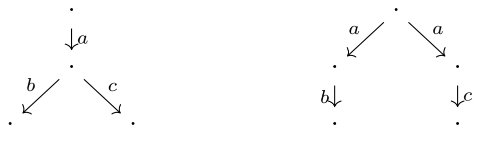
The level of strategies

However, we are not only interested in comparing views and plays, but also *innocent strategies*, which are at the core of game semantics. The square in Figure 5.1 gives a correspondence at the level of plays, but it also yields a tight correspondence between strategies in both approaches. More precisely, it induces an equivalence between innocent strategies in both contexts and shows that this equivalence is compatible with *innocentisation.* Section 5.5 is devoted to proving this result.

To be more precise, there are two notions of innocent strategy in standard game semantics: the first one is a prefix-closed set of views, the second one is a prefix-closed set of plays verifying an extra condition called *innocence.* Both in Tsukada and Ong's approach and in ours, the first notion generalises to *presheaves* on views, which we call *behaviours* and *TO-behaviours.* The second notion generalises to *sheaves* on plays, which we simply call *innocent strategies* and *innocent TO-strategies.* In particular, mere presheaves on plays are possibly non-innocent strategies.

The idea behind this generalisation is the following: a prefix-closed set of views (in, say, $\mathbb{V}_{A,B}$) is a presheaf of booleans $B{:}\mathbb{V}^{op}_{A,B} \to 2$ (where 2 is the ordinal $0 \to 1$ viewed as a category). Similarly, a prefix-closed set of plays is a presheaf $S{:}\mathbb{P}^{op}_{A,B} \to 2$. This presheaf is a sheaf for the Grothendieck topology induced by the embedding of $\mathbb{V}_{A,B}$ into $\mathbb{P}_{A,B}$ when $S(p)$ is accepted if and only if $S(v)$ is accepted for all views $v \to p$. The notion of morphism in $\mathbb{P}_{A,B}$ is slightly non-standard in HON games: morphisms are usually defined as prefix ordering, but they are here defined to satisfy the property that morphisms $v \to p$ correspond to morphisms from a view of p to p (which is not necessarily a prefix of p). The condition of being a sheaf is exactly *innocence*, which states that a player can change its behaviour only according to what they have "seen", i.e., their view, of the play).

However, that is not sufficient to model concurrent strategies, because there may be several different ways to accept a play (or, in other words, a machine may be in several different states after a given trace). The classical example is that of Milner's coffee machines.



Both machines accept exactly the same traces: ε, a, ab, and ac. However, one has a single way of accepting the trace a, after which it still accepts b and c, while the other makes a choice when accepting a whether to accept b or c. The first machine has one way of accepting a, while the second has two. Thus, modelling concurrent strategies correctly requires to know all the different states the strategy can end in after reading a trace: it is not a presheaf of booleans, but a presheaf of sets. Notice that it is now easy to differentiate between the two machines: if we call S_l the strategy of the left-hand machine and S_r that of the right-hand one, we have that $S_l(a)$ is a singleton, while $S_r(a)$ contains two elements, so they indeed are different.

The functors F and $F^{\mathbb{V}}$ give rise to functors $\Delta_F : \widehat{\mathbb{E}(A \vdash B)} \to \widehat{\mathbb{P}_{A,B}}$ and $\Delta_{F^{\mathbb{V}}} : \widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \to \widehat{\mathbb{V}_{A,B}}$, where Δ_f is pre-composition by f^{op}. Since $F^{\mathbb{V}}$ is an equivalence of categories, so is $\Delta_{F^{\mathbb{V}}}$, hence behaviours and TO-behaviours are equivalent.

A strategy is *innocent* (resp. TO-innocent) when it is in the essential image of $\prod_{\mathsf{i}}$ (resp. $\prod_{\mathsf{i}_{TO}}$), where $\prod_f$ denotes right Kan extension along f^{op}. This may also be seen as a sheaf condition stating that an innocent strategy accepts a play if and only if it accepts all the views that can be embedded into that play (see Section 2.2.7). Using the theory of *exact squares* (Section 2.2.6), the square (5.1) provides a categorical explanation of why both induced categories of innocent strategies are equivalent. Indeed, it is exact (Corollary 5.5.2), which means that

$$\begin{tikzcd}
\widehat{\mathbb{V}_{A,B}} \arrow[r, hook, "\prod_{\mathsf{i}_{TO}}"] & \widehat{\mathbb{P}_{A,B}} \\
\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \arrow[u, "\Delta_{F^{\mathbb{V}}}"] \arrow[r, hook, "\prod_{\mathsf{i}}"'] & \widehat{\mathbb{E}(A \vdash B)} \arrow[u, "\Delta_F"']
\end{tikzcd} \tag{5.2}$$

commutes up to isomorphism. In other words, Δ_F turns into an equivalence when restricted to innocent strategies and the *saturation* functors $\prod_{\mathsf{i}} : \mathbb{E}^{\mathbb{V}}_{A \vdash B} \to \mathbb{E}_{A \vdash B}$ and $\prod_{\mathsf{i}_{TO}} : \mathbb{V}_{A,B} \to \mathbb{P}_{A,B}$ that embed behaviours (resp. TO-behaviours) into strategies (resp. TO-strategies) are compatible with this equivalence. This implies that the *innocentisation* functors $\prod_{\mathsf{i}} \circ \Delta_{\mathsf{i}}$ and $\prod_{\mathsf{i}_{TO}} \circ \Delta_{\mathsf{i}_{TO}}$, which turn any strategy into an innocent strategy, are also compatible with this equivalence.

Remark. *Had we wanted to make F an equivalence of categories rather than a mere full embedding, we could easily have imposed an additional condition on our plays akin to alternation in a classical HON-game setting. The point is that we want to compare the purely diagrammatic notion of play with the classical one. And since we obtain an equivalence between both notions of innocent strategies anyway, we feel the result is in fact more convincing.*

Overview

In Section 5.2, we define the base category for HON games in the same way as for the π-calculus, derive categories of views and plays for HON games from it, and give a useful characterisation of these categories as subcategories of coslices of a presheaf category. We then give a slightly informal, but intuitive definition of the embedding of Tsukada and Ong's plays into ours and prove all the desired properties of this embedding in Section 5.3, which is the core result of this chapter. In Section 5.4, we present a formal definition of this embedding and prove all the desired properties. Finally, in Section 5.5, we build on the relationships to show the relationship between plays to give the second result of this paper, which is that both models have equivalent categories of strategies, and that this equivalence is compatible with saturation.

5.2 HON Games as String Diagrams

We first build our model of HON games as string diagrams. We reapply the machinery we have developed in Chapter 4 to build a base category for HON games and derive a pseudo double category of string diagrams representing its plays, as well as categories of views and plays.

5.2.1 Building the Pseudo Double Category

Required: 4.3.3, 2.1.3.
Recommended: ∅.

Sequent calculus and informal description of the game

Like in the case of the π-calculus, our first step is to give an operational description of our language to guide us through steps *(i)–(iii)*. For arena games, the operational description comes in the form of a sequent calculus on arenas (see Sections 2.1.1 and 2.1.3 for details). This is close in spirit to Melliès's work [83] – though the latter takes place in a linear setting. The idea is to understand an arena $A = \sum_i m_i.A_i$ as a logical formula much like $\bigwedge_i \neg A_i$, and to consider the straightforward focalised [8] sequent calculus on these formulas. Remember that the set of roots of A is denoted by $\sqrt{A}$, and that for all m in $\sqrt{A}$, $A \cdot m$ denotes the arena strictly below m.

In our case, the sequent calculus that will guide our construction is:

$$\overset{\Lambda_{(\Gamma\vdash A),m}}{\frac{\Gamma, A\cdot m \vdash}{\Gamma \vdash A}} \qquad \overset{@_{(\Gamma,A,\Delta\vdash),|\Gamma|+1,m}}{\frac{\Gamma, A, \Delta \vdash A\cdot m}{\Gamma, A, \Delta \vdash}} \qquad \overset{\text{CUT}}{\frac{\Gamma \vdash A \qquad \Delta, A, \Delta' \vdash}{\Delta, \Gamma, \Delta' \vdash}} \;, \tag{5.3}$$

where sequents are lists of arenas, with possibly a distinguished arena, written $(A_1, \ldots, A_n \vdash)$ or $(A_1, \ldots, A_n \vdash A)$. Let Γ range over lists of arenas, $|\Gamma|$ denote the length of Γ, and for all $i \in |\Gamma|$, Γ_i the ith arena of Γ. We generalise these notations to sequents: this generalisation is obvious when the sequent is of the form $S = (\Gamma \vdash)$, then $|S| = |\Gamma|$ and $S_i = \Gamma_i$, and when $S = (\Gamma \vdash A)$, we choose $|S| = |\Gamma|$ and $S_i = \Gamma_i$, which may be a bit surprising, but this definition makes statements simpler in our work.

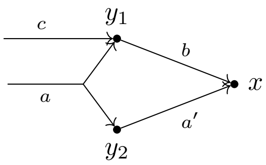
Remark. *The sequent calculus we use here does not make much sense from a logical point of view. It however makes sense when seen in relationship with the following sequent calculus:*

$$\overset{\text{RIGHT}}{\frac{\Gamma, A\cdot m \vdash \quad (\forall m \in \sqrt{A})}{\Gamma \vdash A}} \qquad \overset{\text{LEFT}}{\frac{\Gamma, A, \Delta \vdash A\cdot m}{\Gamma, A, \Delta \vdash}} \qquad \overset{\text{CUT}}{\frac{\Gamma \vdash A \qquad \Delta, A, \Delta' \vdash}{\Delta, \Gamma, \Delta' \vdash}} \;. \tag{5.4}$$

Note that this sequent calculus is a fragment of intuitionistic logic when an arena $A = \sum_{i\in n} m_i.A_i$ is interpreted as $[\![A]\!] = \bigwedge_{i\in n} (\neg\, [\![A_i]\!])$, and proofs in this calculus are proofs of intuitionistic logic.

Positions will be some kind of graphs whose vertices (which we again call players) are labelled by sequents, whose edges (which we again call channels) are labelled by arenas, and such that each player labelled $(\Gamma \vdash A)$ has $|\Gamma|$ incoming edges labelled Γ_i and one outgoing edge labelled A, and similarly for players labelled $(\Gamma \vdash)$.

Example 5.2.1. *For example, the drawing below represents a position with three players: x labelled $(B, A \vdash)$, y_1 labelled $(A, C \vdash B)$, and y_2 labelled $(A \vdash A)$; as well as four edges: a and a' labelled A, b labelled B, and c labelled C (we did not write the label of channels and only their names for readability, but we usually do the opposite and do not write names and only labels).*



We see in this example that our graphs are not exactly graphs, but something slightly more general, for two reasons: edges may have several targets, and they may also be open-ended.

As in our game for the π-calculus, we think of players as agents that may interact through channels. We now want to express the possible interactions in this game, and our sequent calculus again guides us through this step. The CUT rule shows when two players may interact: when the first player's outgoing channel is equal to one of the second player's incoming channels. The interaction between players is governed by the shapes of the Λ and @ rules. In other words, two players can interact according to the Λ and @ rules if and only if they are linked by a CUT rule. Now we just have to show the "shape" of this interaction.

As prescribed by Curry-Howard, we see this interaction as a step in a form of proof of a certain formula. We write $A = \sum_{i \in n} m_i.A_i$ for the arena shared by the two players. Let x by the negative player, labelled $(\Gamma \vdash A)$, and y the positive one, labelled $(\Delta, A, \Delta' \vdash)$; x should provide a proof of $[\![A]\!]$, while y should require such a proof to prove a contradiction. When they interact, y chooses some $i \in n$ and asks x for a proof of $\neg [\![A_i]\!]$, i.e., y now provides a proof of $[\![A_i]\!]$ which x may inspect to prove a contradiction. Therefore, y turns into $(\Delta, A, \Delta' \vdash A_i)$, while x turns into $(\Gamma, A_i \vdash)$.

Moreover, x should not only change into a proof of contradiction assuming $[\![A_i]\!]$, but since y may inspect the proof of $[\![A]\!]$ again, the original x player should also remain in the final position.

Another way to look at this is to see the dynamics of interaction as cut elimination. If we extend our calculus with sharing (we do not describe how this works exactly), then it comes with the following cut elimination rule:

$$\dfrac{\dfrac{\pi}{\Gamma \vdash A} \qquad \dfrac{\dfrac{\pi'}{\Delta_1, A, \Delta_2 \vdash A \cdot m}}{\Delta_1, A, \Delta_2 \vdash}}{\Delta_1', \Gamma, \Delta_2' \vdash} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A} \qquad \dfrac{\pi'}{\Delta_1, A, \Delta_2 \vdash A \cdot m}}{\Delta_1', \Gamma, \Delta_2' \vdash A \cdot m} \qquad \dfrac{\pi_m}{\Gamma, A \cdot m \vdash}}{\Delta_1', \Gamma, \Delta_2' \vdash},$$

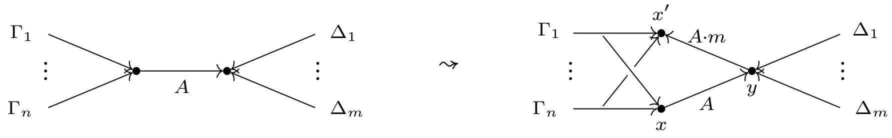
where π is

$$\dfrac{\ldots \qquad \dfrac{\pi_m}{\Gamma, A \cdot m \vdash} \qquad \ldots \qquad (\forall m \in \sqrt{A})}{\Gamma \vdash A}.$$

Indeed, if we see players as the parts of a proof that interact with each other, in the CUT rule above left, we see two players sharing a channel of type A on which they may interact according to π and π' followed by a LEFT rule respectively. After applying the cut elimination rule, we are now in presence of three players that act according to π, π', and π_m respectively. This means that, in the final position of the interaction, there should be three players: a player x labelled $(\Gamma \vdash A)$ that corresponds to the left part of the proof, a player y labelled $(\Delta_1, A, \Delta_2 \vdash A \cdot m)$ that corresponds to the middle part of the proof, and a player x' labelled $(\Gamma, A \cdot m \vdash)$ that corresponds to the right part of the proof.

We see that the left part of the proof is linked to the middle part by a CUT rule on an arena A, while the middle and right parts of the proof are linked by a CUT rule on an arena $A \cdot m$, which did not exist in the initial proof. This means that x and y should share the corresponding channel of type A, while y and x' should share the corresponding channel of type $A \cdot m$. Moreover, Γ is shared between the left and right parts of the proof, so x and x' should share all their channels in Γ. Finally, we may notice that the left part of the proof is π both before and after eliminating the CUT rule, so x should have exactly the same behaviour before and after interacting. This will be imposed by having x "survive" the move, i.e., in such an interaction, the player labelled $(\Gamma \vdash A)$ in the final position of the move is equal to the one in the initial position (and not just another player with the same label).

If we try to picture this, we get that such a move should have for initial position the one on the left-hand side below, and for final position the one on the right-hand side below.



A third and final way to understand this interaction is in terms of interpretation of programs. In our game, players are placeholders for program fragments, and channels are the means through which two program fragments may interact. When we see things this way, positive players (those players that are labelled $(\Gamma \vdash)$) represent program fragments that are currently computing, while negative players (those labelled $(\Gamma \vdash A)$) represent program fragments waiting to be called on. A positive player labelled $(\Gamma \vdash)$ represents a program fragment that has access to resources of type Γ_i, some of which may be continuations. A negative player labelled $(\Gamma \vdash A)$ represents a program fragment that has access to resources of type Γ_i (among which there may be continuations) and provides a value of type A.

Two players x labelled $(\Gamma \vdash A)$ and y labelled $(\Delta, A, \Delta' \vdash)$ interact when the program fragment corresponding to y calls the one corresponding to x.

When this happens, y must stop computing to wait for x's answer, and x starts computing, with a new resource available: the continuation to y to give their answer. We thus see that there is a reversal of roles: x was negative before the interaction and becomes positive afterwards, and *vice versa* for y. Moreover, x should still have access to the resources in Γ, so it should now be labelled $(\Gamma, A \cdot m \vdash)$. Similarly, y should also still have access to the resources it had access to before, so it should be labelled $(\Delta, A, \Delta' \vdash A \cdot m)$, waiting to be called back on $A \cdot m$. Finally, y should be able to call x again later in their execution if they need to, and x should behave "the same" if we want our game to model a pure language, so the original copy of x should still be present in the final position.

Stage *(i)*: positions

Let us now formalise this. We call $\mathbb{L}$ the base category on which the pseudo double category of string diagrams for HON games is built. We first build $\mathbb{L}_1$, which represents positions in our game, as follows:

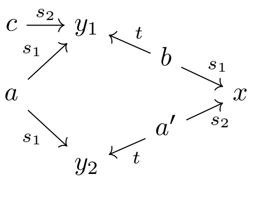
Definition 5.2.2. *Let $\mathbb{L}_1$ be the category freely generated by the graph with*

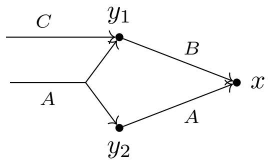
- *a vertex for each arena,*
- *a vertex for each sequent,*
- *for each sequent $S = (A_1, \ldots, A_n \vdash)$ and $i \in n$, an edge $s_i : A_i \to S$,*
- *for each sequent $S = (A_1, \ldots, A_n \vdash A)$, an edge $t : A \to S$, and for all $i \in n$, an edge $s_i : A_i \to S$.*

Example 5.2.3. *The (informal) position of Example 5.2.1 is modelled as the presheaf X defined by:*

- *$X(A) = \{a, a'\}$, $X(B) = \{b\}$, $X(C) = \{c\}$,*
- *$X(A, C \vdash B) = \{y_1\}$, $X(A \vdash A) = \{y_2\}$, $X(B, A \vdash) = \{x\}$,*
- *$X(c) = \varnothing$ otherwise,*
- *$y_1 \cdot s_1 = y_2 \cdot s_1 = a$, $y_1 \cdot s_2 = c$, $y_1 \cdot t = x \cdot s_1 = b$, and $y_2 \cdot t = x \cdot s_2 = a'$,*

Indeed, if we draw its category of elements, we obtain the graph on the left below, which we draw as on the right, which is the desired position.





Stage *(ii)*: selecting spans

Let us now select the spans our moves are built from. For all arenas $A = \sum_{i \in I} m_i.A_i$ and $i \in I$, we will have a span β corresponding to the move sketched in the informal presentation. But in order for the obtained pseudo double category to be fibred, we need to introduce one additional move for each player in the initial position. This makes two moves: the one corresponding to the $\Lambda_{(\Gamma \vdash A),m}$ and the one corresponding to $@_{(\Gamma,A,\Delta\vdash),|\Gamma|,m}$. As announced above, we should enforce the fact that the player x survives the move β (and therefore Λ as well).

Given this, let us now select spans for the Λ, $@$, and β moves.

For all sequents $S = (\Gamma \vdash A)$ and $m \in \sqrt{A}$, consider the span $\mathcal{S}^{\Lambda}_{S,m}$ =

$$\begin{tikzcd} Y^{\Lambda}_{S,m} & Z^{\Lambda}_{S,m} \arrow[l, "w^{\Lambda}_{S,m}"'] \arrow[r, "u^{\Lambda}_{S,m}"] & X^{\Lambda}_{S,m}, \end{tikzcd}$$

where

- $X^{\Lambda}_{S,m} = (\Gamma \vdash A)$,
- $Y^{\Lambda}_{S,m} = (\Gamma \vdash A) \mid (\Gamma, A \cdot m \vdash)$ is the pushout

$$\begin{tikzcd} \sum_{i \in |\Gamma|} \Gamma_i \arrow[r, "{[s_i]_{i \in |\Gamma|}}"] \arrow[d, "{[s_i]_{i \in |\Gamma|}}"'] \arrow[dr, phantom, "\ulcorner", very near end] & (\Gamma, A \cdot m \vdash) \arrow[d, "\mathrm{inr}"] \\ (\Gamma \vdash A) \arrow[r, "\mathrm{inl}"'] & (\Gamma \vdash A) \mid (\Gamma, A \cdot m \vdash) \end{tikzcd}$$

- and $Z^{\Lambda}_{S,m} = (\Gamma \vdash A)$,

with the obvious morphisms $u^{\Lambda}_{S,m} = id_S$ and $w^{\Lambda}_{S,m} = \mathrm{inl}$. The form of the pushout $(\Gamma \vdash A) | (\Gamma, A{\cdot}m \vdash)$ reflects the fact that x and x' should share all their channels in Γ, as explained in the informal introduction.

For all $S' = (\Delta \vdash)$, $i \in |\Delta|$, and $m \in \sqrt{\Delta_i}$, consider the span $\mathcal{S}^{@}_{S',i,m}$ =

$$\begin{tikzcd} Y^{@}_{S',i,m} & Z^{@}_{S',i,m} \arrow[l, "w^{@}_{S',i,m}"'] \arrow[r, "u^{@}_{S',i,m}"] & X^{@}_{S',i,m}, \end{tikzcd}$$

where

- $X^{@}_{S',i,m} = (\Delta \vdash)$,
- $Y^{@}_{S',i,m} = (\Delta \vdash A_i \cdot m)$
- and $Z^{@}_{S',i,m} = \sum_{i \in |\Delta|} \Delta_i$,

with the obvious morphisms.

For all sequents $S = (\Gamma \vdash A)$ and $S' = (\Delta \vdash)$ with $\Delta_i = A$ and $m \in \sqrt{A}$, consider the span $\mathcal{S}^{\beta}_{S,S',i,m}$ =

$$\begin{tikzcd} Y^{\beta}_{S,S',i,m} & Z^{\beta}_{S,S',i,m} \arrow[l, "w^{\beta}_{S,S',i,m}"'] \arrow[r, "u^{\beta}_{S,S',i,m}"] & X^{\beta}_{S,S',i,m}, \end{tikzcd}$$

where

- $X^{\beta}_{S,S',i,m}$ is the pushout

$$\begin{tikzcd}
A \arrow[r, "s_i"] \arrow[d, "t"'] & S' \arrow[d, "\mathrm{inr}"] \\
S \arrow[r, "\mathrm{inl}"'] & X^{\beta}_{S,S',i,m} \arrow[ul, phantom, "\ulcorner", very near start],
\end{tikzcd}$$

- $Y^{\beta}_{S,S',i,m}$ denotes the pushout

$$\begin{tikzcd}
A + (A \cdot m) \arrow[r, "{[s_i, t]}"] \arrow[d, "{[\mathrm{inl} \circ t, \mathrm{inr} \circ s_{|\Delta|+1}]}"'] & (\Gamma \vdash A \cdot m) \arrow[d, "\mathrm{inr}"] \\
S \mid (\Delta, A \cdot m \vdash) \arrow[r, "\mathrm{inl}"'] & Y^{\beta}_{S,S',i,m} \arrow[ul, phantom, "\ulcorner", very near start],
\end{tikzcd}$$

- and $Z^{\beta}_{S,S',i,m} = (S + \sum_{j \neq i} \Delta_j)$,

with the obvious morphisms.

Stage *(iii)*: augmenting the base category

At last, we augment our base category $\mathbb{L}_1$ with new objects and morphisms that model moves, just like in the case of the π-calculus.

Definition 5.2.4. *Let $\mathbb{L}$ consist of $\mathbb{L}_1$, plus:*

- *For all sequents $S = (\Gamma \vdash A)$ and $m \in \sqrt{A}$, an object $\Lambda_{S,m}$ with maps $S \xrightarrow{t} \Lambda_{S,m} \xleftarrow{s} (\Gamma, A \cdot m \vdash)$ such that $t \circ s_i = s \circ s_i$ for all $i \in |\Gamma|$,*
- *For all sequents $S = (\Delta \vdash)$, $i \in |\Delta|$, and $m \in \sqrt{\Delta_i}$, an object $@_{S,i,m}$ with maps $S \xrightarrow{t} @_{S,i,m} \xleftarrow{s} (\Delta \vdash A_i \cdot m)$ such that $t \circ s_i = s \circ s_i$ for all $i \in |\Delta|$,*
- *For all sequents $S = (\Gamma \vdash A)$ and $S' = (\Delta \vdash)$, with $i \in |\Delta|$ such that $\Delta_i = A$ and $m \in \sqrt{A}$, an object $\beta_{S,S',i,m}$ with maps $\Lambda_{S,m} \xrightarrow{\lambda} \beta_{S,S',i,m} \xleftarrow{@} @_{S',i,m}$ such that*

$$\lambda \circ t \circ t = @ \circ t \circ s_i \qquad \textit{and} \qquad \lambda \circ s \circ s_{|\Delta|+1} = @ \circ s \circ t.$$

For all sequent $S = (\Gamma \vdash A)$ and m in $\sqrt{A}$, we have by Yoneda a morphism $t: X^{\Lambda}_{S,m} \to \Lambda_{S,m}$. Moreover, we get $s': Y^{\Lambda}_{S,m} \to \Lambda_{S,m}$ by universal property of pushout in:

$$\begin{tikzcd}
\sum_{i \in |\Gamma|} \Gamma_i \arrow[r, "{[s_i]_{i \in |\Gamma|}}"] \arrow[d, "{[s_i]_{i \in |\Gamma|}}"'] & (\Gamma, A \cdot m \vdash) \arrow[d, "\mathrm{inr}"] \arrow[ddr, bend left, "s"] & \\
(\Gamma \vdash A) \arrow[r, "\mathrm{inl}"'] \arrow[drr, bend right, "t"'] & Y^{\Lambda}_{S,m} \arrow[ul, phantom, "\ulcorner", very near start] \arrow[dr, dashed, "s'"] & \\
 & & \Lambda_{S,m}.
\end{tikzcd}$$

Proposition 5.2.5. *For all sequents $S = (\Gamma \vdash A)$ and move m in $\sqrt{A}$, the square*

$$\begin{tikzcd}
Z^{\Lambda}_{S,m} \arrow[r, "w^{\Lambda}_{S,m}"] \arrow[d, "u^{\Lambda}_{S,m}"'] & Y^{\Lambda}_{S,m} \arrow[d, "s'"] \\
X^{\Lambda}_{S,m} \arrow[r, "t"'] & \Lambda_{S,m}
\end{tikzcd}$$

is a pullback.

Proof. A simple check. □

There are of course similar propositions for $@_{S,i,m}$ and $\beta_{S,S',i,m}$.

We may now choose the signature describing HON games by choosing a functor from $\mathbb{L}_{|\geq 2}$ to $\mathsf{Cospan}(\widehat{\mathbb{L}})$. In fact, for each valid tuple (S, S', i, m), $\mathbb{L}_{|\geq 2}$ locally looks like the poset

$$\begin{tikzcd}
& \beta_{S,S',i,m} & \\
\Lambda_{S,m} \arrow[ur, "\lambda"] & & @_{S',i,m} \arrow[ul, "@"']
\end{tikzcd}$$

viewed as a category. We define our functor $\mathbb{L}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{L}})_H$ to map this to

$$\begin{tikzcd}
Y^{\Lambda}_{S,m} \arrow[r, "\mathrm{inl}"] \arrow[d, hook] & Y^{\beta}_{S,S',i,m} \arrow[d, hook] & Y^{@}_{S',i,m} \arrow[l, "\mathrm{inr}"'] \arrow[d, hook] \\
\Lambda_{S,m} \arrow[r, "\lambda"] & \beta_{S,S',i,m} & @_{S',i,m} \arrow[l, "@"'] \\
X^{\Lambda}_{S,m} \arrow[r, "\mathrm{inl}"'] \arrow[u, hook] & X^{\beta}_{S,S',i,m} \arrow[u, hook] & X^{@}_{S',i,m}. \arrow[l, "\mathrm{inr}"] \arrow[u, hook]
\end{tikzcd} \tag{5.5}$$

We may easily check that this assignment yields a functor.

Definition 5.2.6. *Let* $\mathsf{S}_{HON} : \mathbb{L}_{|\geq 2} \to \mathsf{Cospan}(\widehat{\mathbb{L}})_H$ *denote the obtained functor.*

5.2.2 Categories of Views and Plays

Required: 5.2.1, 4.4.4.
Recommended: ∅.

We may now construct the pseudo double category $\mathbb{D}_{HON}$ based on the signature S_{HON} using the machinery from Chapter 4. We thus get a pseudo double category of concurrent traces that represent HON games.

We may then prove:

Lemma 5.2.7. $\mathbb{D}_{HON}$ *is fibred.*

Proof. By Corollary 4.4.40, it suffices to show that S_{HON} is persistent, monolithic, and separated, which is routine. □

From this, we get for each position X a category $\mathbb{E}(X)$ of plays over X, as defined in Definition 4.3.30 and which may concretely be characterised as the category with:

- as objects, all plays $U : Y \mathrel{-\!\!\!\bullet\!\!\!\to} X$,

- as morphisms from $U\colon Y \mathrel{-\!\!\bullet\!\!\!\to} X$ to $U'\colon Y' \mathrel{-\!\!\bullet\!\!\!\to} X$, all double cells α as below left, quotiented by the equivalence relation generated by relating α on the left to $\alpha \circ (U \bullet \gamma)$ on the right:

$$\begin{tikzcd}
Z \arrow[r] \arrow[d, "W"', "\bullet" marking] & Y' \arrow[dd, "U'", "\bullet" marking] \\
Y \arrow[d, "U"', "\bullet" marking] \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & \\
X \arrow[r, equal] & X
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
Z' \arrow[r] \arrow[dr, "W'"', "\bullet" marking] \arrow[r, phantom, shift right=4ex, "\overset{\gamma}{\Rightarrow}"] & Z \arrow[r] \arrow[d, "W", "\bullet" marking] & Y' \arrow[dd, "U'", "\bullet" marking] \\
& Y \arrow[d, "U"', "\bullet" marking] \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & \\
& X \arrow[r, equal] & X.
\end{tikzcd}$$

Remark. *Notice that, by Lemma 4.2.1, any α as above whose middle component is 1D-injective automatically has a 1D-pullback bottom square. Moreover, if the middle component is 1D-injective, then the top component also is.*

As explained in Section 4.3.4, composition is defined using cartesian cells. The composition of α and β below is $\beta \circ (\alpha \bullet \gamma)$, where γ is the cartesian lifting of s.

$$\begin{tikzcd}
Z'' \arrow[r, dashed] \arrow[d, dashed, "\bullet" marking] \arrow[dr, phantom, "\overset{\gamma}{\Rightarrow}"] & Z' \arrow[r, "s'"] \arrow[d, "W'", "\bullet" marking] & Y'' \arrow[ddd, "U''", "\bullet" marking] \\
Z \arrow[r, "s"] \arrow[d, "W"', "\bullet" marking] & Y' \arrow[dd, "U'", "\bullet" marking] \arrow[r, phantom, "\overset{\beta}{\Longrightarrow}"] & \\
Y \arrow[d, "U"', "\bullet" marking] \arrow[r, phantom, "\overset{\alpha}{\Rightarrow}"] & & \\
X \arrow[r, "r"'] & X \arrow[r, "r'"'] & X.
\end{tikzcd}$$

Composition defined this way indeed possesses an identity (the identity cell) and is associative (because cartesian cells compose both horizontally and vertically).

To relate to traditional HON game semantics, we need to define categories of views and plays that are the counterparts of the traditional categories $\mathbb{P}_{A,B}$ and $\mathbb{V}_{A,B}$ (see Section 2.1.3 for definitions). For this reason, we now restrict our attention to particular shapes of initial positions. The counterpart to $\mathbb{P}_{A,B}$ in our case is simply the category $\mathbb{E}(A \vdash B)$.

Remark. *The construction may look too involved, since we end up restricting our attention to plays over $(A \vdash B)$, that is, with only one player. For example, β moves can never occur in a play over $(A \vdash B)$. Indeed, for such a move to happen, the outgoing edge of one player must be equal to one of the incoming edges of another player. But we cannot find this pattern in the initial position, and positions where this pattern does not appear are stable under Λ and @ moves, so a β move can never be played.*

However, this machinery is built in the same way as our other string diagrammatic models, and it allows us to treat the multi-party positions that will appear in plays uniformly, and with the same intuitions as in playground models. The β move also makes sense in larger positions, for example when trying to define interaction sequences in string diagrams. In this framework, we may define interaction sequences just like plays, but with a different initial position: an interaction sequence on the arena triple (A, B, C) is a play on the position $(A \vdash B \vdash C)$, which is defined as the pushout

$$\begin{tikzcd}
B \arrow[r, "t"] \arrow[d, "s_1"'] & (A \vdash B) \arrow[d, "\mathrm{inl}"] \\
(B \vdash C) \arrow[r, "\mathrm{inr}"'] & (A \vdash B \vdash C). \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

(This definition is however unsatisfactory for technical reasons, and we need to adapt the definition of the base category for it to be meaningful.)

In particular, we use this category in Chapter 6 to study fine invariants of interaction sequences in HON games.

We now need to give a counterpart to the notion of view. But fibred pseudo double categories are inspired from the notion of *playgrounds* [50, 29], a categorical gadget that is in turn inspired from game semantics and come with a notion of view. In the setting of playgrounds, there are special moves called *basic moves*, which track only one player, and views are defined as composites of such moves. The idea is that, if the behaviour of a player only depends on its view (which is the basic idea of *innocence*), then its behaviour cannot depend on how other players play.

In our case, the idea is the same, but this exact approach fails on two points.

- Since there is a duality in HON game semantics between Opponent and Proponent, who usually play alternatively, and we are only interested in Proponent (because we want to describe the behaviour of the program, and not that of the environment), only composites of basic moves of even length should be views.

- We do not have all the basic moves that we need to define views this easily. Indeed, one of the playground axioms, the *axiom of views*, basically states that, for all moves $M\colon Y \mathrel{\longrightarrow\!\!\!\!\!\bullet\,} X$ and player $y\colon Y$ in the final position of M, there should exist a cell

$$\begin{tikzcd}
d \arrow[r, "y"] \arrow[d, dashed, "v^{y,M}"' , "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha^{y,M}}{\dashrightarrow}"] & Y \arrow[d, "M", "\bullet" marking] \\
d^{y,M} \arrow[r, dashed, "x^{y,M}"'] & X,
\end{tikzcd}$$

where $v^{y,M}$ is a basic move or an equivalence and $x^{y,M}\colon d^{y,M} \to X$ is a player. As the reader can see, this means that basic moves should have players as their initial and final positions, and that for any move and player in its final position, there should be a basic move that corresponds to what that player "sees" of the move. But, if we take the move $\Lambda_{(\Gamma\vdash A),m}$ and the player $\mathrm{inr}\colon (\Gamma, A\cdot m \vdash) \to Y^{\Lambda}_{S,m}$, there is no way to fill the diagram

$$\begin{tikzcd}
(\Gamma, A\cdot m \vdash) \arrow[r, "\mathrm{inr}"] \arrow[d, dashed, "v"', "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha}{\dashrightarrow}"] & Y^{\Lambda}_{S,m} \arrow[d, "\Lambda_{S,m}", "\bullet" marking] \\
d \arrow[r, dashed, "x"'] & (\Gamma \vdash A).
\end{tikzcd}$$

What we do at this point in our other models is to define a new move, say $\lambda_{(\Gamma\vdash A),m}$, that only contains the player labelled $(\Gamma, A\cdot m \vdash)$ in its final position. However, if we do that, then the channel labelled A in the initial position is not present in the final position, which breaks persistence, and thus our proof of fibredness.

Because of this, we should find an alternative definition of view that follows the same idea: a view should only follow one player.

Definition 5.2.8. *For any play $U{:}\,Y \twoheadrightarrow (A \vdash B)$, we define the binary relation $\prec_U$ on all moves $m{:}\,\mu \to U$ of U (where $\mu \in \mathbb{L}_{|\geq 2}$) by $m \prec_U m'$ if and only if $m \cdot s = m' \cdot t$. When $m \prec_U m'$, we say that m'* causally depends *on m. Furthermore, we omit the subscript when clear from context.*

A preview *is a play $U{:}\,Y \twoheadrightarrow (A \vdash B)$ such that the reflexive, transitive closure $\prec^*$ of $\prec_U$ is a total order on the moves of U.*

A view *is a preview of positive, even length (i.e., it is the composite of a positive, even number of moves).*

In other words, a preview is a play that does not contain any β moves and that can be written as a composite $M_1 \bullet \ldots \bullet M_n$ of moves in a unique way (up to isomorphism of moves). This means that we may assign a number to each move, according to its place in the previous sequence, and that the kth move has to occur before the $k+1$th. In particular, no two moves may occur "in parallel", or, if we decompose U as $U_1 \bullet Q \bullet U_2$ with Q a quasi-move (recall Definition 4.4.37), then Q is actually a move. It also implies that U is an alternation of Λ and @ moves, and that the $k+1$th move is played by the unique player *created* by the kth move (i.e., that player who is in the final position of the kth move, but not in its initial position).

Our counterpart to $\mathbb{V}_{A,B}$ is:

Definition 5.2.9. *Let $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ be the full subcategory of $\mathbb{E}(A \vdash B)$ spanning views.*

Let us also give a useful lemma on views:

Lemma 5.2.10. *Let $V{:}\,Z \twoheadrightarrow (A \vdash B)$ be a view isomorphic to a composite of a play $U{:}\,Y \twoheadrightarrow (A \vdash B)$ and a move $M{:}\,Z' \twoheadrightarrow Y$. Then U is also a view.*

Proof. Let $(l, k, id_{(A \vdash B)}){:}\,U \bullet M \to V$ be the isomorphism. Because k is an isomorphism, we have that $\prec_{U \bullet M}$ and $\prec_V$ have the same structure: all moves m_V in V are in the image of k, and $m \prec_{U \bullet M} m'$ if and only if $k(m) \prec_V k(m')$. Since V is a view, we know that $\prec_V$ is of the form $k(m_1) \prec_V \ldots \prec_V k(m_n)$, where m_1, $\ldots$, m_n are all the moves in $U \bullet M$. Therefore, we have $m_1 \prec_{U \bullet M} \cdots \prec_{U \bullet M} m_n$. Now, it suffices to notice that $\prec_{U \bullet M}$ is an extension of $\prec_U$, and that adding the move M can only add pairs $m \prec m'$ in which m is a move from U and m' is the move from M, so m' is necessarily m_n, so $\prec_U^*$ is a total order. □

We finally give the notions of strategy that are associated to this game. As mentioned in this chapter's introduction, there are actually two notions of strategies, just like in game semantics.

Definition 5.2.11. *An $(A \vdash B)$-*behaviour *(or simply a* behaviour*) is a presheaf over $\mathbb{E}^{\mathbb{V}}(A \vdash B)$.*

Definition 5.2.12. *An $(A \vdash B)$-*strategy *(or simply a* strategy*) is a presheaf over $\mathbb{E}(A \vdash B)$. A strategy is* innocent *when it is in the essential image of $\prod_{\mathsf{i}}$, where i is the inclusion of $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ into $\mathbb{E}(A \vdash B)$.*

5.2.3 Characterisations of Views and Plays

Required: 5.2.2.
Recommended: ∅.

Before relating our categories of views and plays with Tsukada and Ong's, we digress a little in this section to establish the announced characterisation of $\mathbb{E}(A \vdash B)$ and $\mathbb{E}^{\mathbb{V}}(A \vdash B)$, as subcategories of the coslice $(A \vdash B)/\widehat{\mathbb{L}}$.

Definition 5.2.13. *Let $\mathbb{E}'(A \vdash B)$ denote the subcategory of $(A \vdash B)/\widehat{\mathbb{L}}$ spanning morphisms $t\colon (A \vdash B) \to U$ for which there exists a play $Y \xrightarrow{s} U \xleftarrow{t} (A \vdash B)$, and 1D-injective morphisms (Definition 4.3.10) between them.*

Let $(\mathbb{E}^{\mathbb{V}})'(A \vdash B)$ denote the full subcategory of $\mathbb{E}'(A \vdash B)$ spanning views.

There is an obvious candidate functor $\mathcal{U}\colon \mathbb{E}(A \vdash B) \to \mathbb{E}'(A \vdash B)$ mapping $Y \xrightarrow{s} U \xleftarrow{t} (A \vdash B)$ to t and a morphism

$$\begin{tikzcd}
Z \arrow[rrr] \arrow[dr] & & & Y' \arrow[dd] \\
& W \arrow[dr] & & \\
Y \arrow[ur] \arrow[dr] & & U \bullet W \arrow[r, "\alpha"] & U' \\
& U \arrow[ur] & & \\
(A \vdash B) \arrow[ur] \arrow[rrr, equal] & & & (A \vdash B) \arrow[uu]
\end{tikzcd}$$

from $\langle U \rangle$ to $\langle U' \rangle$ (remember Notation 4.4.14) to the composite $U \to (U \bullet W) \xrightarrow{\alpha} U'$.

Lemma 5.2.14. *$\mathcal{U}$ is compatible with the equivalence relation $\sim$ (defined in Notation 4.3.27) and yields a functor $\mathcal{U}\colon \mathbb{E}(A \vdash B) \to \mathbb{E}'(A \vdash B)$.*

Proof. To show that $\mathcal{U}$ is compatible with $\sim$, we simply show that, for all situations as in (4.9), $(\langle W \rangle, \alpha)$ and $(\langle W' \rangle, \alpha \circ (u \bullet \gamma))$ have the same image through $\mathcal{U}$. Given a diagram as the solid part of

$$\begin{tikzcd}
Z' \arrow[r] \arrow[dr] & Z \arrow[rrr] \arrow[dr] & & & Y' \arrow[dd] \\
& W' \arrow[r, "\gamma"] \arrow[dr] & W \arrow[dr] & & \\
Y \arrow[ur] \arrow[urr] \arrow[dr] & & U \bullet W' \arrow[r, dashed, "U \bullet \gamma"] & U \bullet W \arrow[r, "\alpha"] & U' \\
& U \arrow[ur] \arrow[urr] & & & \\
X \arrow[ur] \arrow[rrrr, equal] & & & & X, \arrow[uu]
\end{tikzcd}$$

we get $U \bullet \gamma$ by universal property of pushout, and the images through $\mathcal{U}$ of the two morphisms we are interested in are

$$U \to U \bullet W \xrightarrow{\alpha} U' \qquad \text{and} \qquad U \to U \bullet W' \xrightarrow{U \bullet \gamma} U \bullet W \xrightarrow{\alpha} U',$$

which are equal by construction of $U \bullet \gamma$. Functoriality of the obtained assignment is straightforward. □

The rest of this section is devoted to proving:

Theorem 5.2.15. *$\mathcal{U}$ is an equivalence, and thus restricts to an equivalence at the level of views $\mathcal{U}^{\mathbb{V}} \colon \mathbb{E}^{\mathbb{V}}(A \vdash B) \to (\mathbb{E}^{\mathbb{V}})'(A \vdash B)$.*

Since $\mathcal{U}$ is surjective on objects by definition, we just need to prove that the underlying functor to $(A \vdash B)/\widehat{\mathbb{L}}$ is faithful and that its image on homsets precisely spans 1D-injective morphisms.

This is in fact an easy consequence of:

Lemma 5.2.16. *Assume given plays $u \colon Y \mathrel{-\!\!\bullet\!\!\to} (A \vdash B)$ and $u' \colon Y' \mathrel{-\!\!\bullet\!\!\to} (A \vdash B)$, and a 1D-injective morphism $h \colon \mathcal{U}(u) \to \mathcal{U}(u')$.*

There exists a quasi-move $w \colon Z \mathrel{-\!\!\bullet\!\!\to} Y$ and a morphism $\alpha \colon (u \bullet w) \to u'$ in $\mathbb{D}(\mathsf{S}_{HON})_H$ such that $h = (U \to U \bullet W \xrightarrow{\alpha} U')$, which is minimal in the sense that, for any (w', α') such that $h = (U \to U \bullet W' \xrightarrow{\alpha'} U')$, there exists a unique $\gamma \colon w \to w'$ such that α decomposes as $\alpha' \circ (u \bullet \gamma)$, as in

$$\begin{tikzcd}
Z \arrow[r] \arrow[dr, "w"', ""{name=W}] & Z' \arrow[rr, "s"] \arrow[d, "w'", ""{name=WP, swap}] & & Y' \arrow[dd, "u'"] \\
& Y \arrow[rr, phantom, "\overset{\alpha'}{\Longrightarrow}"] \arrow[d, "u"'] & & \\
& (A \vdash B) \arrow[rr, equal] & & (A \vdash B).
\arrow[Rightarrow, from=W, to=WP, "\gamma"]
\end{tikzcd}$$

In the rest of the proof, we show how to build (w, α) and how to factor it through any (w', α') as above. The idea is that positive players (those of the form $(\Gamma \vdash)$) are always present in the final position (they can only play Λ moves, and they are present in the final positions of those moves), while negative players (those of the form $(\Gamma \vdash A)$) are in the final position of a play if and only if they never play (because they are not present in the final positions of @ moves). For $u \bullet w$ to have a map α to u', the final position Z of w must be contained in that of u', which means that all negative players in Z must also exist in Y'. To ensure this, we define w' to play @ moves for all the negative players of w' that are not in the final position of u'.

In order to prove this, we need to analyse final positions of plays as follows. Given a position $X \in \widehat{\mathbb{L}}$, recalling Terminology 4.3.9, let us call a channel an *input* when it occurs as $x \cdot s_i$, for some player $x \in X(S)$ for some sequent S, and an *output* when it occurs as $x \cdot t$. In general positions, channels may be both inputs and outputs, but in coproducts of sequents, each channel is one or the other, but not both. Let us generalise this situation:

Definition 5.2.17. *A position is* polar *when each of its channels is either an input or an output but not both.*

Non-polar positions either have disconnected channels (which are neither inputs nor outputs) or channels which are both inputs and outputs.

From this point on until the end of the proof of Lemma 5.2.16, we will implicitly use the fact that β moves never occur in the plays we consider, which holds because we only consider plays starting from polar positions. In particular, we will use notations such as $m \cdot t$ or $m \cdot s$ for arbitrary moves m, which is ill-defined in general, but well-defined for Λ and @ moves.

Lemma 5.2.18. *Any polar position admits a surjective and 1D-injective map from a coproduct of sequents, given by the counit of the comonad of Proposition 4.4.10.*

Proof. The counit is clearly 1D-injective, and it is surjective because all channels are connected to at least one player (otherwise, they would not have a polarity). □

Lemma 5.2.19. *For all plays $u{:}\, Y \mathrel{-\!\!\bullet\!\!\!\to} (A \vdash B)$, Y is polar.*

Proof. This follows from the more general fact that, for any $u{:}\, Y \mathrel{-\!\!\bullet\!\!\!\to} X$ with polar X, Y is polar, which is proved by induction on the length of u. □

Definition 5.2.20. *An* interface *is a position consisting only of channels.*

Any polar position X comes with a canonical (up to isomorphism) monic map $I_X + O_X \hookrightarrow X$ from some coproduct of two interfaces I_X and O_X, surjective (hence iso) in dimension 0, such that I_X covers inputs and O_X covers outputs. Note that all maps preserve polarity, i.e., any $h{:}\, X \to Y$ maps positive channels to positive channels and negative ones to negative ones, but they may also add polarities to some channels, so, e.g., a positive channel may be mapped to a channel that is both positive and negative.

Notation 5.2.21. *We call any map as in Lemma 5.2.18 a* polar cover *of the given position. We fix a global choice of such polar covers, which, for any polar X, we denote by $\mathrm{Pl}(X) = (\sum_{i \in P_X} X_i^P) + (\sum_{i \in N_X} X_i^N) \overset{\varepsilon_X}{\twoheadrightarrow} X$, where each X_i^P is a* positive *sequent, i.e., one of the form $(\Gamma \vdash)$, and each X_i^N is a* negative *sequent, i.e., one of the form $(\Gamma \vdash A)$, and P_X and N_X are the numbers of positive and negative players in X, respectively.*

Finally, by Lemma 4.4.28, any play u on $\mathrm{Pl}(X)$ "descends" to a play $(\varepsilon_X \cdot u){:}\, (\varepsilon_X \cdot Y) \mathrel{-\!\!\bullet\!\!\!\to} X$, as the pushout

$$\begin{tikzcd}
id^\bullet_{I_{\mathrm{Pl}(X)}+O_{\mathrm{Pl}(X)}} \arrow[r] \arrow[d] \arrow[dr, phantom, "\ulcorner", very near end] & id^\bullet_{I_X+O_X} \arrow[d] \\
u \arrow[r, "\alpha^u_X"'] & \varepsilon_X \cdot u.
\end{tikzcd}$$

We now need a further observation on final positions. These are *a priori* associated to some play, but in fact we may pose:

Definition 5.2.22. *The* final position $\upharpoonleft U$ *of a presheaf $U \in \widehat{\mathbb{L}}$ is the smallest subpresheaf of U containing all channels and negative players, as well as all positive players x for which there exists no move m with $m \cdot t = x$. We deem the elements of $\upharpoonleft U$* final *in U.*

Accordingly, a morphism $h{:}\, Y \to U$ with Y a position, is called final *if and only if it is isomorphic to $\upharpoonleft U \to U$ (in $\widehat{\mathbb{L}}/U$).*

Intuitively, the final position retains only those positive players who haven't yet played any move.

Lemma 5.2.23. *For all plays $u{:}Y \mathrel{-\!\!\bullet\!\!\!\to} (A \vdash B)$, s_u is final.*

Proof. By induction on u. □

Our next step will have to do with fullness of $\mathcal{U}$. It will rely on the following notion:

Definition 5.2.24. *For any play $u{:}Y \mathrel{-\!\!\bullet\!\!\!\to} (A \vdash B)$, a 1D-injective morphism $h{:}X \to U$ (remember Notation 4.4.14) is P-*ample *if and only if for all S, i, r and $m \in U(@_{S,i,r})$, if $m \cdot t \in \mathrm{Im}(h)$ then $m \cdot s \notin \mathrm{Im}(h)$.*

The idea behind P-ample morphisms is that such a morphism $X \to U$ only takes a "slice" of the play at some point in time, and so all players in X have causally independent images in U. (This is only a rough idea, and there may a player causally dependent on another in a P-ample morphism.)

Definition 5.2.25. *For any play $u{:}Y \mathrel{-\!\!\bullet\!\!\!\to} (A \vdash B)$, final players of U are called* survivors*, while non-final players are called* doomed*. The set of survivors of U is denoted by* $\mathrm{Surv}(U)$ *and the set of doomed players by* $\mathrm{Doom}(U)$*.*

For any $h{:}X \to U$, we reflect the decomposition of $\mathrm{pl}(U)$ *into survivors and doomed players as* $\mathrm{pl}(X) = \mathrm{Surv}(X) \uplus \mathrm{Doom}(X)$*.*

Of course, all doomed players are positive.

Lemma 5.2.26. *For any play u and moves m_1 and m_2 in U, if $m_1 \cdot s = m_2 \cdot s$, then $m_1 = m_2$.*

Proof. By induction on u. □

Notation 5.2.27. *For any play $u{:}Y \mathrel{-\!\!\bullet\!\!\!\to} X$, let $\lceil u \rceil$ denote the cospan $Y \xrightarrow{s_u} U = U$.*

Lemma 5.2.28. *Assume given any polar position X, play $u{:}Y \mathrel{-\!\!\bullet\!\!\!\to} (A \vdash B)$, and P-ample morphism $h{:}X \to U$.*

There exist quasi-seeds (vertical morphisms which are either seeds or identities) $M_i{:}Y_i \mathrel{-\!\!\bullet\!\!\!\to} X_i^P$ for all $i \in P_X$ and a 1D-injective α_h as in

$$\begin{tikzcd}
(\sum_{i \in P_X} Y_i + \sum_{j \in N_X} X_j^N) \arrow[rr] \arrow[dd, "M_h"', "\bullet" marking] & & \varepsilon_X \cdot (\sum_{i \in P_X} Y_i + \sum_{j \in N_X} X_j^N) \arrow[rr] \arrow[dd, "\varepsilon_X \cdot M_h", "\bullet" marking] & & Y \arrow[dd, "\lceil u \rceil", "\bullet" marking] \\
 & \overset{\alpha_X^{M_h}}{\Longrightarrow} & & \overset{\alpha_h}{\Longrightarrow} & \\
(\sum_{i \in P_X} X_i^P + \sum_{j \in N_X} X_j^N) \arrow[rr, two heads, "\varepsilon_X"'] & & X \arrow[rr, "h"'] & & U,
\end{tikzcd}$$

where bottom squares are all 1D-pullbacks, $M_h = (\sum_{i \in P_X} M_i + \sum_{j \in N_X} X_j^N)$ and, on the right, the cospan $\lceil u \rceil$ is viewed as a vertical morphism in $\mathsf{Cospan}(\widehat{\mathbb{L}})$.

Proof. Let $X^N = \sum_{j \in N_X} X_j^N$ and $X^P = \sum_{i \in P_X} X_i^P$. By Lemma 5.2.23, any negative player x in X uniquely corresponds to some negative player x' in Y, mapped to $h(x)$ by s_u. This yields a cell

$$\begin{tikzcd}
X^N \arrow[r] \arrow[d, equals, "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha^N}{\Longrightarrow}"] & Y \arrow[d, "\bullet" marking, "\lceil u \rceil"] \\
X^N \arrow[r] & U.
\end{tikzcd}$$

Now, for any positive survivor x in X (over some sequent S_x), if we define $Y_x = S_x$, there is a cell

$$\begin{tikzcd}
Y_x \arrow[r] \arrow[d, equals, "\bullet" marking] \arrow[dr, phantom, "\overset{\alpha_x}{\Longrightarrow}"] & Y \arrow[d, "\bullet" marking, "\lceil u \rceil"] \\
S_x \arrow[r, "\ulcorner h(x) \urcorner"'] & U
\end{tikzcd}$$

analogous to the one above.

Finally, for any doomed $x \in X(S_x)$, let $x' = h(x)$ denote its image in $U(S_x)$. Because x is doomed, there exists a (unique) move $m \in U(@_{S_x,i,r})$ for some i and r, such that $m \cdot t = x'$. Let $(M_x\colon Y_x \mathrel{-\!\!\bullet\!\!\to} S_x) = \mathsf{S}_{HON}(@_{S_x,i,r})$ denote the seed of m. By definition of S_{HON}, Y_x is a negative sequent, and we let $y = m \cdot s \in U(Y_x)$. By the same argument as above, y has a unique antecedent y' in Y and so there exists a cell

$$\begin{tikzcd}
Y_x \arrow[r, "\ulcorner y' \urcorner"] \arrow[d, "\bullet" marking, "M_x"'] \arrow[dr, phantom, "\overset{\alpha_x}{\Longrightarrow}"] & Y \arrow[d, "\bullet" marking, "\lceil u \rceil"] \\
S_x \arrow[r, "\ulcorner h(x) \urcorner"'] & U.
\end{tikzcd}$$

By copairing all these cells, we obtain a cell $\alpha^0_h\colon M_h \to \lceil u \rceil$ in $\mathsf{Cospan}(\widehat{\mathbb{L}})$, which decomposes as desired by universal property of pushout and the fact that $id^\bullet_{I_U+O_U}$ is the universal interface with a map to U (so $I_X + O_X \to U$ factors through $I_U + O_U$). It remains to prove that all involved maps are 1D-injective and that all bottom squares are 1D-pullbacks.

Let us first show that α^0_h is 1D-injective. In dimension 2, if $m, m'\colon @_{S,i,r} \to M_h$ are two moves such that $\alpha^0_h m = \alpha^0_h m'$, then $(\alpha^0_h m) \cdot t = (\alpha^0_h m') \cdot t$, so $\alpha^0_h(m \cdot t) = \alpha^0_h(m' \cdot t)$, but $m \cdot t$ and $m' \cdot t$ have antecedents in $X^N + X^P$, so this contradicts 1D-injectivity of $h\varepsilon_X$. In dimension 1, the set of players of M_h is isomorphic to the coproduct of players of X^N, X^P, and Y^N, where Y^N denotes $\sum_{x \in \mathrm{Doom}(X)} Y_x$, the set of players created by the $@_{S,i,r}$ moves in M_h. Since $h\varepsilon_X$ is 1D-injective, players in the image of X^N and X^P cannot be equated by α^0_h. Players in Y^N and X^P cannot be equated because they lie over different objects (those in Y^N are over negative sequents, while those in X^P are over positive ones). By construction, players in Y^N cannot be equated by α^0_h: by Lemma 5.2.26, in a play, if $m \cdot s = m' \cdot s$, then $m = m'$. So the only possibility for α^0_h not to be 1D-injective is for a player in Y^N to be equated with a player in X^N. Let us call y^N and x^N these two players. By construction of M_h, we know that there is a player x^P in the image of X^P and a move m in M_h such that $y^N = m \cdot s$ and $x^P = m \cdot t$. But then, $h(\varepsilon_X(x^N)) = \alpha^0_h(y^N) = \alpha^0_h(m) \cdot s$ and $h(\varepsilon_X(x^P)) = \alpha^0_h(m) \cdot t$, which contradicts P-ampleness of h.

The left-hand square has a 1D-pullback bottom square by Lemma 4.4.28 and the fact that $id^\bullet_{I_{\mathrm{Pl}(X)}+O_{\mathrm{Pl}(X)}} \to id^\bullet_{I_X+O_X}$ is bijective in dimensions > 0. The large rectangle has a 1D-pullback bottom square by Lemma 4.2.2. Finally, the right-hand square has a 1D-pullback bottom square by Lemma 4.2.3. □

We have three further handy lemmas:

Lemma 5.2.29. *For any horizontal morphism $h: X' \to X$ and play $u: Y \twoheadrightarrow X$, there exists at most one cell $id^{\bullet}_{X'} \to u$ with bottom border h.*

For any $@_{S,i,r} \in \mathrm{ob}(\mathbb{L})$, morphism $h: S \to X$, and play $u: Y \twoheadrightarrow X$, there exists at most one cell $\mathsf{S}_{HON}(@_{S,i,r}) \to u$ with bottom border h.

Proof. The first proof is by monicity of $Y \to U$. For the second, because $X \to U$ and $Y \to U$ are monic, any cell $\alpha: \mathsf{S}_{HON}(@_{S,i,r}) \to u$ is uniquely determined by its middle component $@_{S,i,r} \to U$. But, by Yoneda, that component itself is determined the image of $m = id_{@_{S,i,r}}$ (which is the move element in $\mathsf{S}_{HON}(@_{S,i,r})$). But this is in fact uniquely determined by the image of $t \in \mathsf{y}_{@_{S,i,r}}$ (the player element), so it is uniquely determined by $h(id_S)$. Indeed, by an easy induction on u, there is at most one $m' \in U(@_{S,i,r})$ such that $m' \cdot t = h(x)$. □

Lemma 5.2.30. *For any play u and player x in U, either x is in the image of t_u or there exists a move m such that $x = m \cdot s$. Dually, either x is in the image of s_u or there exists m in $U(@_{S,i,r})$, for some S, i, r, such that $x = m \cdot t$.*

Proof. By induction on u. □

We now define a relation analogous to $\prec$ (Definition 5.2.8), but on players:

Definition 5.2.31. *For all plays u and players x and y in u, let $x \prec_u y$ if and only if there exists m such that $x = m \cdot t$ and $y = m \cdot s$. As before, we omit the subscript when clear from context.*

Lemma 5.2.32. *For all morphisms $h: \mathfrak{U}(u) \to \mathfrak{U}(u')$ in $\mathbb{E}'(A \vdash B)$, if $h(x) \prec_{u'} h(y)$ with x positive, i.e., over some $(\Gamma \vdash)$, then $x \prec_u y$.*

Proof. Let m' witness $h(x) \prec_{u'} h(y)$. By Lemma 5.2.30, we have $h(y) \notin \mathrm{Im}(t_{u'})$, so $y \notin \mathrm{Im}(t_u)$, and hence there exists $m \in U$ such that $m \cdot s = y$, again by Lemma 5.2.30. Now, by naturality of h, we have $h(m) \cdot s = h(y)$ so $h(m) = m'$ by Lemma 5.2.26. Again by naturality of h, we get $h(m \cdot t) = m' \cdot t = h(x)$, so by 1D-injectivity of h we obtain $x = m \cdot t$ and hence $x \prec_u y$. □

Lemma 5.2.16 now follows:

Proof of Lemma 5.2.16. Given u, u' and h, let $k = h \circ s_u$. This morphism is P-ample. Indeed, consider any $m' \in U'(@_{S,i,r})$ with $k(x) = m' \cdot t$, and $k(y) = m' \cdot s$. Then we have $k(x) \prec_{u'} k(y)$ so by Lemma 5.2.32, there exists m witnessing $s_u(x) \prec_u s_u(y)$, so $s_u(x) \in \mathrm{Im}(s_u)$ and $m \cdot t = s_u(x)$, which, by Lemma 5.2.30, is a contradiction.

Since k is P-ample, by Lemma 5.2.28, we get a cell $(l, \alpha_k, k): w \to \lceil u' \rceil$ with $w = \varepsilon_Y \cdot M_k: Z \twoheadrightarrow Y$ a quasi-move. This in particular ensures that the square

$$\begin{tikzcd}
Y \arrow[r] \arrow[d] & W \arrow[d] \\
U \arrow[r, "h"'] & U'
\end{tikzcd}$$

commutes. By universal property of pushout, this induces a unique morphism $\alpha{:}\, u \bullet w \to u'$ such that $\alpha \circ inj_U^W = h$ and $\alpha \circ inj_W^U = \alpha_k$ (where $inj_U^W{:}\, U \hookrightarrow U \bullet W$ and $inj_W^U{:}\, W \hookrightarrow U \bullet W$ are the pushout injections). Let us show that α is 1D-injective by showing that $W \smallsetminus Y$ and $U \smallsetminus Y$ have disjoint images in U' in dimensions > 0. In dimension 1, if $\alpha_k(x_W) = h(x_U)$ for some players x_W of $W \smallsetminus Y$ and x_U of U, then by construction of W, x_W is a negative player, hence so is x_U, and therefore x_U is in Y by Lemma 5.2.23. In dimension 2, if $\alpha_k(m_W) = h(m_U)$ for some moves m_W of W and m_U of U, then by construction of W, m_W must be such that $m_W \cdot t = t_w(y)$ for some player y in Y, and m_W is also necessarily an @ move, and thus, so must m_U. Now, $h(m_U \cdot t) = h(m_U) \cdot t = \alpha_k(m_W) \cdot t = \alpha_k(m_W \cdot t) = \alpha_k(t_w(y)) = k(y) = h(s_u(y))$, so by 1D-injectivity of h, $m_U \cdot t = s_u(y)$, which is a contradiction by Lemma 5.2.30. To show that the bottom square of $\alpha{:}\, u \bullet w \to u'$ is a 1D-pullback, it suffices to notice that α is 1D-injective and the bottom component is an identity, so the square is a 1D-pullback by Lemma 4.2.1.

It remains to show that the pair (w, α) is minimal. Consider thus any (w', α') such that $\alpha' \circ inj_U^{W'} = h$.

By Lemma 5.2.30, a positive player $x \in U(S)$ in the image of t_u is doomed if and only if there is a move $m \in U(@_{S,i,r})$, for some i and r, such that $x = m \cdot t$.

Now, let us observe that, by construction of W, for any positive player x of Y, $t_w(x) \in \mathrm{Doom}(W)$ if and only if $k(x) \in \mathrm{Doom}(U')$.

But, for any player x in Y, if $k(x)$ is doomed in U' then $t_{w'}(x)$ is doomed in W' (otherwise, $t_{w'}(x)$ has an antecedent in Z' by Lemma 5.2.30, and thus $k(x)$ has an antecedent in Y', which is a contradiction, again by Lemma 5.2.30).

This entails that $M_k \to U'$ lifts through $\alpha' \circ inj_{W'}^U$, as desired, uniquely by Lemma 5.2.29. But then $w \to U'$ also lifts through α' by universal property of pushout, uniquely because $\alpha_Z^{M_k}$ is epi. □

Proof of Theorem 5.2.15. Lemma 5.2.16 easily entails that the image of $\mathcal{U}$ in $(A \vdash B)/\widehat{\mathbb{L}}$ on homsets spans 1D-injective morphisms. Faithfulness follows from minimality of the constructed pair (w, α). □

5.3 The Level of Plays: Intuition

We now want to define the functor $F{:}\, \mathbb{P}_{A,B} \to \mathbb{E}(A \vdash B)$ and show that it is a full embedding and that it restricts to an equivalence $F^{\mathbb{V}}{:}\, \mathbb{V}_{A,B} \to \mathbb{E}^{\mathbb{V}}(A \vdash B)$ on views. For reasons we explain at the end of this section, the arguments used here are not as formal as one would expect, so this whole section is here more to give the reader intuition about F than a formal construction and proofs of its properties, which are delayed to Section 5.4.

5.3.1 Illustration on an Example

Required: 5.2.3.
Recommended: ∅.

To give some intuition about how this embedding works, we apply it to an example, illustrated in Figure 5.1. This embedding is decomposed into two steps: the first one maps TO-plays to some form of proof trees, and the second

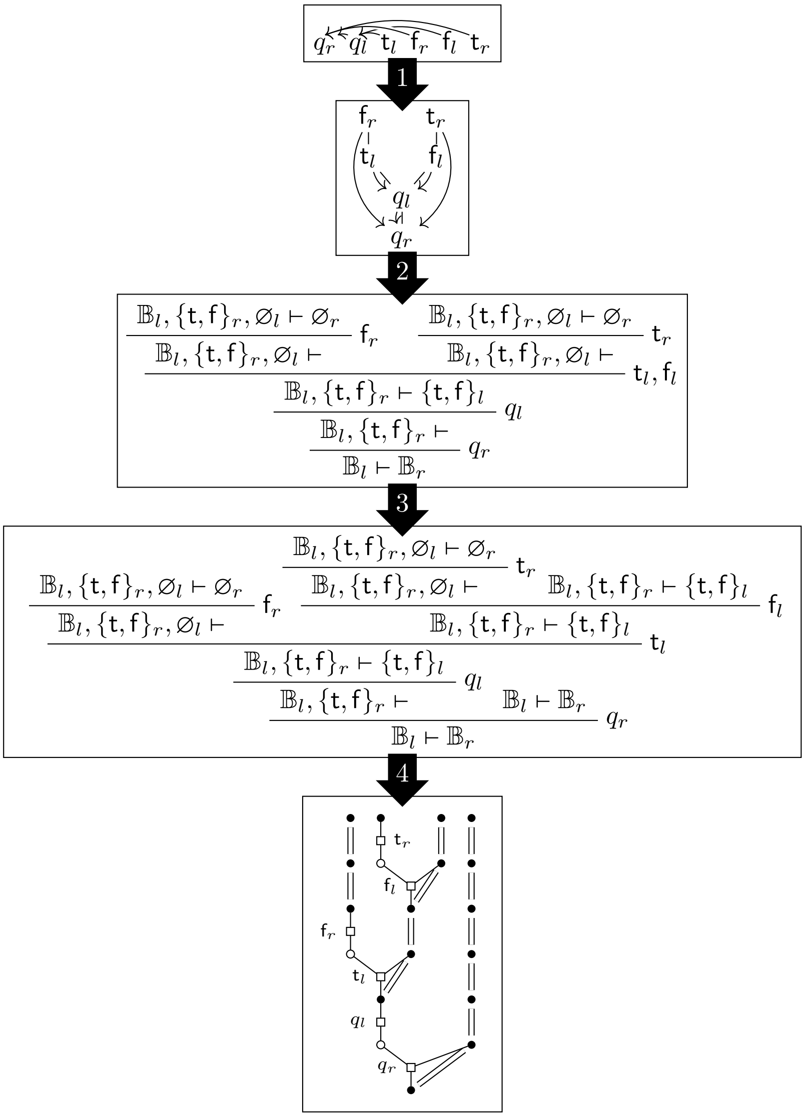
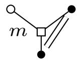
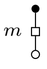
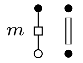

q_r q_l t_l f_r f_l t_r
1
f_r t_r
t_l f_l
q_l
q_r
2
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r$ f_r
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash$
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r$ t_r
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash$
$\mathsf{t}_l, \mathsf{f}_l$
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash \{\mathsf{t},\mathsf{f}\}_l$ q_l
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash$ q_r
$\mathbb{B}_l \vdash \mathbb{B}_r$
3
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r$ t_r
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r$ f_r
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash$
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash \{\mathsf{t},\mathsf{f}\}_l$ f_l
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash$
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash \{\mathsf{t},\mathsf{f}\}_l$ t_l
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash \{\mathsf{t},\mathsf{f}\}_l$ q_l
$\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash$
$\mathbb{B}_l \vdash \mathbb{B}_r$ q_r
$\mathbb{B}_l \vdash \mathbb{B}_r$
4
t_r
f_l
f_r
t_l
q_l
q_r

Figure 5.1: The big picture

one maps these proof trees to plays as string diagrams. In our example, we further refine both steps: TO-plays are first mapped to *P-view trees* [16], then to proof trees, then to a linearised form of the same proof tree, and finally to a string diagram.

In our example, we study a play s on $(\mathbb{B}_l, \mathbb{B}_r)$, the pair of two boolean arenas, where l and r are only present to show in which copy of $\mathbb{B}$ moves are played. We show how s may be mapped to a string diagram P, explain why this mapping is functorial, and how it restricts to an equivalence on views.

Our plays are as defined in Section 5.2, but, for the sake of readability, we omit channels when drawing positions. We thus need to have a way to distinguish between positive and negative players. We depict the positive ones as ○ and negative ones as ● . The only two moves that may occur in a play on $(A \vdash B)$ are built from the seeds $\Lambda_{S,m}$ and $@_{S,i,m}$, which are depicted as  and  respectively, with their initial positions at the bottom and the final ones at the top, but where we omit channels. We can then draw moves by drawing a seed and the rest of the position, as in  , which is a move on a position with two players, with the left player playing an @ move. A *play* can then be represented as a vertical pasting of moves.

A morphism of plays f (i.e., a morphism in $\mathbb{D}(\mathsf{S}_{HON})_H$) is then just like an injective morphism of graphs whose vertices are players and whose edges are moves: it sends players and moves labelled c to players and moves with the same label and respects the graph structure of the play. It should also respect the initial and final positions in the sense that the image of the source's initial and final positions should be contained in the target's initial and final positions. A morphism $f{:}U \to U'$ in $\mathbb{E}(A \vdash B)$ is then just an inclusion of the graph representing U into that representing U' that respects the initial position.

As a first step (Step 1 in Figure 5.1), we map our example play s to its P-view tree [16], a tree whose branches are the TO-views of s. There are two types of "arrows" in this tree: the ones depicted as proper arrows, which correspond to justification pointers in s, and the ones that create the tree structure, which correspond to the view of each move, as defined inductively:

- if m is an O-move, then in the tree it is the child of its predecessor in the sequence,
- if m is a P-move, then in the tree it is the child of its justifier in the sequence.

Since, in the tree structure, all O-moves point to their predecessors, P-view trees can only branch at odd depth.

We then map this P-view tree (Step 2) to a partial proof tree (i.e., a proof tree whose branches may be left unfinished) in a sequent calculus based on arenas. Partial proof trees in this sequent calculus represent *explorations* of proofs in sequent calculus (5.4) (what we call exploration is basically a sub-tree of the whole proof), so it is not very surprising that these partial proof trees are equivalent to string diagrams. By definition of the sequent calculus these proof trees will be built on, they may branch arbitrarily, but only at odd depth, just like P-view trees. This notion of play comes with a notion of view, which are simply branches in this sequent calculus. Even though both notions are trees and look alike (the structures are clearly similar), there are some subtle differences: the P-view tree is labelled by moves, while the proof tree is labelled by sequents of arenas, and the P-view tree contains pointers, while the proof tree does not. A little bit of work has to be done to prove that this mapping is a full embedding and that it restricts to an equivalence on views. We prove this for the composite of Steps 1 and 2 in Section 5.3.3.

Steps 3 and 4 are conceptually simple, but somewhat tricky to formalise, so we do not detail them in the rest of the chapter. Step 3 maps the proof tree, which may branch with an arbitrary degree, to a similar tree that only branches in a binary way. Let us call the result of Step 3 a *sequential* proof tree (in the sense that we have sequentialised some of the tree structure). To do this, we change the rules on which our proof trees are built. Here, the rules that have a positive sequent as conclusion are the same as in the previous proof trees, but the rules that have a negative sequent as conclusion have necessarily two premises: a positive one like in the previous proof trees, and a copy of the conclusion. Sequentialising a proof tree is now completely obvious: if a node has n premises, we just apply the sequential rule n times. There is a choice to be made on the order in which we sequentialise the rules, but all these choices lead to isomorphic plays in the end. For example, we have decided to linearise the tree by applying t_l first, and then f_l in our example, but the reverse ordering would clearly yield the same string diagram. Step 4 is completely direct: all the rules of the proof tree now exactly correspond to moves in our game, so we just mimic the structure of the proof tree. Nodes in the tree become players with the same label, and deduction rules become moves in our game. Here again, since the tree is a branching structure, we may choose to apply the moves in different orders, but all choices lead to isomorphic plays. In our example, we have chosen to play t_l, f_r, f_l, and t_r in that order, but we could have played f_r at any point after t_l (and all the resulting plays are isomorphic).

To show that this mapping reduces to an equivalence when restricted to views on both sides, we should first say what our views look like when drawn like the bottom play in Figure (5.1). They are simply non-branching plays, in the sense that, for all Λ moves (the only moves that can possibly branch), only the created player may play in the rest of the play. For example, the play depicted at the bottom of Figure (5.1) is not a view because, both players in the final position of the t_l move play (f_r and f_l respectively). Equivalently, since all equal signs are mathematical equalities in the presheaf that represents the play, we may collapse the whole structure along these equal signs. A play is then a view if and only if this collapse is linear, i.e., it never branches. Now, we can map views in our setting back to TO-views by reading moves from bottom to top in our view (this is unambiguous because views do not branch). We explain in Section 5.3.3 how to recover pointers from the structure of a play, and how this implies that the sequence defined by taking moves from bottom to top in a view is actually a TO-view.

Finally, let us briefly mention the case of morphisms. Morphisms of TO-plays are injective functions that respect views (in the sense that it “maps views to views”). Through Step 1, they are mapped fully and faithfully to injective morphisms of rooted trees, i.e., those injective morphisms of trees that map its source's root to its target's root. Through Step 2, they are mapped fully and faithfully to inclusions of proof trees, which is the notion of morphism of proof trees. This is not surprising since P-view trees are mapped to proof trees that have exactly the same structure and the notions of morphisms are similar in both settings. Through Steps 3 and 4, inclusions of proof trees are mapped fully and faithfully to morphisms in $\mathbb{E}^{\mathbb{V}}(A \vdash B)$. This is once again not surprising because both notions of morphisms are similar, in the sense that they can both be seen as morphisms of trees.

5.3.2 From String Diagrams to Proof Trees

Required: 5.3.1.
Recommended: $\varnothing$.

In this section, we build another model of HON games based on proof trees and show that it has equivalent categories of views and plays as our string diagrammatic model.

The sequent calculus these proof trees are built on is:

$$\frac{\dots \quad \Gamma, A \cdot m(i) \vdash \quad \dots \quad (\forall i \in n)}{\Gamma \vdash A}\,\textsc{right} \qquad \frac{\Gamma, A, \Delta \vdash A \cdot m}{\Gamma, A, \Delta \vdash}\,\textsc{left} \tag{5.6}$$

where the RIGHT rule can be applied with any $m{:}\, n \to \sqrt{A}$ for a positive n and the LEFT rule can be applied with any $m \in \sqrt{A}$. Notice how, apart from the absence of a CUT rule, the sequent calculus (5.4) we have introduced at the beginning differs only slightly from this one: there, in the RIGHT rule, m must always be a bijection. Moreover, while the objects of interest are "complete" trees in the first case, the objects of interest here are "incomplete" trees. This once again corresponds to the fact that we are interested in explorations of proofs, and not proofs themselves. Therefore, some branches of the proof may not be explored (hence the fact that m need not be surjective) and others may be explored more than once (hence m need not be injective).

An *S-tree* is a partial proof tree (i.e., a proof tree whose branches may be left unfinished) whose conclusion is S. For all sequents S, the counterpart of $\mathbb{E}(S)$ is the category $\mathbb{T}(S)$ of S-trees and morphisms of such, which are inclusions, both in width and depth, of S-trees, defined inductively by:

- the empty S-tree has exactly one morphism to all other S-trees,
- if, for all i in n, T_i is a $(\Gamma, A \cdot m_i \vdash)$-tree, and for all i in m, T'_i is a $(\Gamma, A \cdot m'_i \vdash)$-tree, then the set of morphisms from $\dfrac{T_1 \quad \dots \quad T_n}{\Gamma \vdash A}$ to $\dfrac{T'_1 \quad \dots \quad T'_m}{\Gamma \vdash A}$ is the disjoint union, for all injective $g{:}\, n \to m$ such that $m'_{g(i)} = m_i$, of $\prod_{i \in n} \mathbb{T}(\Gamma, A \cdot m_i \vdash)(T_i, T'_{g(i)})$,
- if T and T' are $(\Gamma, A, \Delta \vdash A \cdot m)$-trees, then the set of morphisms from $\dfrac{T}{\Gamma, A, \Delta \vdash}$ to $\dfrac{T'}{\Gamma, A, \Delta \vdash}$ is $\mathbb{T}(\Gamma, A, \Delta \vdash A \cdot m)(T, T')$.

Notice that morphisms treat the premises of a $(\Gamma \vdash A)$ node as if they were unordered.

We also need a counterpart of $\mathbb{E}^{\mathbb{V}}(S)$, which is given by S-branches:

Definition 5.3.1. *An S-*branch *is a non-branching S-tree (i.e., an S-tree whose* RIGHT *rules are all unary) of positive, even depth. Let $\mathbb{B}(S)$ be the full subcategory of $\mathbb{T}(S)$ spanning S-branches.*

Remark. *Representing plays as trees is reminiscent of Boudes's work [16], and our trees are indeed close to his P-view trees (it may be shown that a slight variant of this notion is equivalent to ours), but there are some differences. The first one is that his notion of morphism of trees is different. In particular, the functor from TO-plays to trees is not full with his definition of morphism. The second one is that we allow branches of odd depth, when he does not. Both these differences are easily remediable, but there are two more fundamental differences: the nodes of his trees are labelled by moves in an arena while ours are labelled by sequents of arenas, and his P-view trees contain pointers while our trees do not. Some work must therefore be done to know whether the two notions coincide.*

Example 5.3.2. *The tree in the middle of Figure 5.1 (between Steps 2 and 3) is an example of $(\mathbb{B},\mathbb{B})$-tree. Here is an example of $(A \vdash B)$-tree:*

$$\frac{A, B \cdot m \vdash \qquad A, B \cdot m \vdash}{A \vdash B}\, m, m \ .$$

None of these trees are branches (they both branch at some point). Note that, while the first tree intuitively represents a TO-play, the other one does not, as it lacks alternation: it is as if Opponent had played m twice in a row. This example also shows that morphisms treat premises as if they were unordered, in the sense that there are two morphisms from that tree to itself (one that maps both branches to themselves and one that swaps them).

Here is an example of $(A \vdash B)$-branch:

$$\frac{\dfrac{A, B \cdot m \vdash A \cdot m'}{A, B \cdot m \vdash}\, m'}{A \vdash B}\, m \ .$$

We will later need this technical result:

Proposition 5.3.3. *$\mathbb{T}(A, B \vdash C)$ and $\mathbb{T}(A + B \vdash C)$ are isomorphic.*

This is a conceptually simple result, and we only sketch the proof.

Proof. We want to show by induction that $\mathbb{T}_N(A, B \vdash C)$ and $\mathbb{T}_N(A + B \vdash C)$ are isomorphic, where $\mathbb{T}_N(S)$ is the category of S-trees of height at most S. This is enough to prove the main result, because $\mathbb{T}(S)$ is the colimit of the $\mathbb{T}_N(S)$'s, and colimits of isomorphic diagrams are isomorphic.

But we first need to generalise the induction hypothesis. For all integers n, m and maps $\gamma : n \to m$, we define $\gamma(A_1, \ldots, A_n)$ as the m-tuple whose kth component is $\sum_{\gamma(i)=k} A_i$. We generalise this notation to sequents in the obvious way: $\gamma(\Gamma \vdash A) = (\gamma(\Gamma) \vdash A)$ and $\gamma(\Gamma \vdash) = (\gamma(\Gamma) \vdash)$. It is now easy to show that for all integers n and m and all maps $\gamma : n \to m$ and sequent S of length n, $\mathbb{T}_N(S) \cong \mathbb{T}_N(\gamma(S))$. □

We now show that $(A \vdash B)$-trees and branches are just an alternative way of representing plays and views:

Lemma 5.3.4. *$\mathbb{T}(A \vdash B)$ and $\mathbb{E}(A \vdash B)$ are equivalent, and so are $\mathbb{B}(A \vdash B)$ and $\mathbb{E}^{\mathbb{V}}(A \vdash B)$, and these equivalences are such that*

$$\begin{tikzcd}
\mathbb{B}(A \vdash B) \arrow[r] \arrow[d] & \mathbb{T}(A \vdash B) \arrow[d] \\
\mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[r] & \mathbb{E}(A \vdash B)
\end{tikzcd}$$

commutes.

The intuition behind this statement is that sequents in the proof tree correspond to players who exist at some point in the play, and that deduction rules correspond to what such players can do during the play. The trees are meant to be read bottom up: at the beginning of a play in $\mathbb{E}(A \vdash B)$, there is only one player of type $(A \vdash B)$ present in the play. Because that player is negative, they are present in the final position of any move they play, and there is no causal dependence between the different moves they can play, so they can be seen as being played in parallel, independently from one another, which corresponds to the RIGHT rule with a family I of cardinal n if the player plays n different moves. On the contrary, positive players are never present in the final positions of any move they play, so they may only play once, hence the form of the LEFT rule, which allows for a single move to be played.

Proof. By Theorem 5.2.15, the lemma is equivalent to showing that $\mathbb{T}(A \vdash B)$ and $\mathbb{B}(A \vdash B)$ are equivalent to the subcategories of the coslices of Theorem 5.2.15 (which are denoted $\mathbb{E}'(S)$ and $(\mathbb{E}^{\mathbb{V}})'(S)$ for any sequent S) and that the desired diagram commutes. We actually show a slightly more general result, by replacing $(A \vdash B)$ above by any sequent S.

The basic idea is very simple: if U is a play and T is an S-tree that is equivalent to it (for the equivalence that will be defined below), then nodes of T correspond exactly to players of U; edges of T exactly to moves of U; and an edge e connects a node x to its parent y in T exactly when the player that corresponds to y plays the move that corresponds to e, resulting in the creation of a new player that corresponds to x.

For any sequent S, S-trees may be interpreted as elements of $S/\widehat{\mathbb{L}}$ inductively like so:

- the empty S-tree is interpreted as $id\colon S \to S$,
- if T is a $(\Gamma, A, \Delta \vdash A\cdot m)$-tree interpreted as $f\colon (\Gamma, A, \Delta \vdash A\cdot m) \to U$, then $\dfrac{T}{\Gamma, A, \Delta \vdash}$ is interpreted as $\widetilde{t}$ defined by universal property of pushout in

$$\begin{tikzcd}
(\Gamma, A, \Delta \vdash) \arrow[r, "t"] \arrow[dr, dashed, "\widetilde{t}"'] & @_{(\Gamma,A,\Delta),|\Gamma|+1,m} \arrow[d] & (\Gamma, A, \Delta \vdash A\cdot m) \arrow[l, "s"'] \arrow[d, "f"] \\
& \widetilde{U} \arrow[ur, phantom, "\lrcorner", very near start] & U, \arrow[l]
\end{tikzcd}$$

- if T_i's are $(\Gamma, A\cdot m(i) \vdash)$-trees interpreted as $f_i\colon (\Gamma, A\cdot m(i) \vdash) \to U_i$, then $\dfrac{T_1 \quad \ldots \quad T_n}{\Gamma \vdash A}$ is interpreted as the arrow $(\Gamma \vdash A) \to U'$ resulting from the wide pushout of $\widetilde{t_1}, \ldots, \widetilde{t_n}$:

$$\begin{tikzcd}
(\Gamma \vdash A) \arrow[r, "\widetilde{t_1}"] \arrow[d, "\widetilde{t_n}"'] \arrow[dr, dashed, "\widetilde{t}"] \arrow[d, phantom, shift left=3, "\iddots"] & \widetilde{U_1} \arrow[d] \\
\widetilde{U_n} \arrow[r] & \widetilde{U}, \arrow[ul, phantom, very near start, "\ulcorner"]
\end{tikzcd}$$

where $\widetilde{t_i}$ is defined by universal property of pushout in:

$$\begin{tikzcd}
(\Gamma \vdash A) \arrow[r, "t"] \arrow[dr, dashed, "\widetilde{t_i}"'] & \Lambda_{(\Gamma \vdash A), m(i)} \arrow[d] & (\Gamma, A \cdot m(i) \vdash) \arrow[l, "s"'] \arrow[d, "f_i"] \\
 & \widetilde{U_i} \arrow[u, phantom, very near start, "\ulcorner"] & U_i . \arrow[l]
\end{tikzcd}$$

The interpretation of morphisms of S-trees is a morphism in the coslice $S/\widehat{\mathbb{L}}$, and it is defined by induction on the S-tree structure:

- if the interpretation of T is $f\colon S \to U$, the only morphism from the empty S-tree to T is sent to f (remember that the interpretation of the empty tree is id_S),
- if $T \to T'$ is sent to $\alpha\colon f \to f'$, we get a morphism from the interpretation of $\dfrac{T}{\Gamma, A, \Delta \vdash}$ to that of $\dfrac{T'}{\Gamma, A, \Delta \vdash}$ by universal property of pushout in

$$\begin{tikzcd}
(\Gamma, A, \Delta \vdash) \arrow[r, "t"] \arrow[dr, "\widetilde{t}"'] \arrow[ddr, "\widetilde{t'}"'] & @_{(\Gamma, A, \Delta), |\Gamma|+1, m} \arrow[d] & (\Gamma, A, \Delta \vdash A \cdot m) \arrow[l, "s"'] \arrow[d, "f"] \\
 & \widetilde{U} \arrow[d, dashed] \arrow[u, phantom, very near start, "\ulcorner"] & U \arrow[l] \arrow[d, "\alpha"] \\
 & \widetilde{U'} & U', \arrow[l]
\end{tikzcd}$$

- if $T_i \to T'_{g(i)}$ is sent to $\alpha_i\colon f_i \to f'_i$, then we get $\widetilde{\alpha_i}\colon \widetilde{U_i} \to \widetilde{U'_{g(i)}}$ as above, and a morphism from the interpretation of $\dfrac{T_1 \quad \dots \quad T_n}{\Gamma \vdash A}$ to that of $\dfrac{T'_1 \quad \dots \quad T'_m}{\Gamma \vdash A}$ by universal property of wide pushout in

$$\begin{tikzcd}
S \arrow[rr] \arrow[dd] \arrow[dr, equal] \arrow[dr, phantom, shift right=3, "\iddots"] & & \widetilde{U_1} \arrow[dd, dashed] \arrow[dr] & & \widetilde{U'_k} \arrow[dddd] \\
 & S \arrow[urrr] \arrow[rr] \arrow[dd] \arrow[dddl] \arrow[dr, phantom, "\iddots"] & & \widetilde{U'_{g(1)}} \arrow[dddr] & \\
\widetilde{U_n} \arrow[rr] \arrow[dr] & & \widetilde{U} \arrow[ddrr, dashed] & & \\
 & \widetilde{U'_{g(n)}} \arrow[drrr] & & & \\
\widetilde{U'_l} \arrow[rrrr] & & & & \widetilde{U'},
\end{tikzcd}$$

where k, …, l are the indices of m that are not in the image of g.

This interpretation is functorial.

The interpretation of the LEFT rule corresponds to an @ move, and the interpretation of the RIGHT rule to an n-composite of Λ moves. Therefore, the interpretation of an S-tree is a composite of moves, so it is a play, hence the image of the interpretation is contained in $\mathbb{E}'(S)$.

Conversely, any play in $\mathbb{E}(S)$ may be interpreted as an S-tree. The interpretation of a play P is defined by induction on the number of moves P is composed of. The construction maintains the invariants that all nodes in the S-tree correspond to exactly one player of the same type in P, and that a node x is a child of y if and only if the player corresponding to x is created by a move played by y in P. The empty play (composite of 0 moves) is interpreted as the empty S-tree, and if P is composite $Q \bullet M$ where Q is a play and M a move [1], and Q is interpreted as the S-tree on the left below, where the node S' corresponds to the player who plays M, then P is interpreted as on the right below, where S'' is the resulting sequent (i.e., if $S' = (\Gamma \vdash A)$ and M is a $\Lambda_{S',m}$ move, then $S'' = (\Gamma, A \cdot m \vdash)$, and if $S' = (\Gamma \vdash)$ and M is an $@_{S',i,m}$ move, then $S'' = (\Gamma \vdash \Gamma_i \cdot m))$, and the new S'' node in the tree corresponds to the player created by M.

$$\dfrac{T_1 \quad \dots \quad T_n \qquad \dfrac{T'_1 \quad \dots \quad T'_m}{S'}}{T} \qquad\qquad \dfrac{T_1 \quad \dots \quad T_n \qquad \dfrac{T'_1 \quad \dots \quad T'_m \quad S''}{S'}}{T}$$

Even though this looks wrong at first sight in the case where S' is of the form $(\Gamma \vdash)$ (because then the tree could possibly branch on such sequents, which is forbidden by the structure of our trees), we know that such a player may only play at most once in a play, so m must be equal to 0 in that case, so there is no contradiction.

Moreover, in the case of plays whose initial positions consist of a single player, a morphism in $U \to V$ in $\mathbb{E}(S)$ can be reduced to an injective mapping of moves of U to those of V that preserves both causality (if a player x is created by a move m and plays a move m', then the image of m' must be played by a player created by the image of m) and the initial player (the image of all moves played by the initial player of U must be played by the initial player of V). Therefore, a morphism of plays $U \to V$ is equivalent to a mapping of edges in the interpretation of U to those in the interpretation of V that respects causality (if an edge is under another in the source tree, then its image must be under the other's image in the target tree) and initiality (the image of all edges that start from the root must start from the root), i.e., they are exactly morphisms of S-trees. This interpretation is also functorial.

It can easily be seen that both composites of these two interpretations (the one from $\mathbb{T}(S)$ to itself and the one from $\mathbb{E}'(S)$ to itself) are naturally isomorphic to identities: for any of the two interpretations, the number of nodes in the tree is equal to the number of players in the corresponding play (and they have the same type), the number of edges of the tree to that of moves in the play (and they have again the same type), and their structure (what we called "initiality" and "causality" above) is preserved. Therefore, the image of a tree T through the composite of both interpretations has the same number of nodes and edges

[1] There may be several such decompositions, but they all result in the same interpretation.

as as T, and these nodes and edges are structured exactly the same in both trees, so they are isomorphic (and similarly for plays).

Therefore, the two interpretations form an equivalence between $\mathbb{T}(S)$ and $\mathbb{E}'(S)$, and therefore $\mathbb{T}(S)$ and $\mathbb{E}(S)$ are equivalent.

Moreover, the definition of views ensures exactly that views are linear, in the sense that a player cannot play more than one move. It is then easy to show that, if a player cannot play more than one move, then its interpretation must be a branch, and conversely that, in the interpretation of a branch, each player must play at most once. Therefore, the equivalence restricts to an equivalence between $\mathbb{B}(S)$ and $\mathbb{E}^{\mathbb{V}}(S)$, hence the result. □

5.3.3 From Proof Trees to Justified Sequences

Required: 5.3.2.
Recommended: ∅.

Now that we have introduced our model based on proof trees and shown that it is equivalent to that based on string diagrams in the sense of Lemma 5.3.4, we want to show that this tree model is related to Tsukada and Ong's in a similar sense. Namely, we want to construct a full embedding from $\mathbb{P}_{A,B}$ to $\mathbb{T}(A \vdash B)$ and then show that it restricts to an equivalence $F^{\mathbb{V}} \colon \mathbb{V}_{A,B} \to \mathbb{B}(A \vdash B)$. By combining both results, we thus get a full embedding from $\mathbb{P}_{A,B}$ to $\mathbb{E}(A \vdash B)$ that restricts to an equivalence on the respective categories of views. This will then yield a further correspondence between categories of strategies.

Let us first build $F(s)$ by induction on s. If s is empty, $F(s)$ is simply the empty $(A \vdash B)$-tree. Otherwise, s can be written as a sum [2] of threads $s = \sum_{i \in n} t_i$ (recall the definition of of thread from Definition 2.1.53). Now, each thread t_i is of the form $m_i^1 m_i^2 s_i$ for moves m_i^1, m_i^2 and a play s_i, and a slight generalisation of a result at the start of Section 3.5 in [97] gives:

Proposition 5.3.5. *For all arenas A and B, if $\mathsf{T}_{A,B}$ is the full subcategory of $\mathbb{P}_{A,B}$ spanning threads, $\chi \colon (mm')/\mathsf{T}_{A,B} \to \mathbb{P}_{C,C \cdot m'}, (mm's) \mapsto s$ is an isomorphism, where $C = A + B \cdot m$.*

Each s_i can therefore be mapped to a $(C_i \vdash C_i \cdot m_i^2)$-tree recursively, where $C_i = A + B \cdot m_i^1$. By Proposition 5.3.3, this gives an $(A, B \cdot m_i^1 \vdash C_i \cdot m_i^2)$-tree $\widetilde{F}(s_i)$, which can in turn be composed with the LEFT and RIGHT rules to give $F(s)$ as below:

$$\dfrac{\cdots \qquad \dfrac{\widetilde{F}(s_i)}{A, B \cdot m_i^1 \vdash}\; m = m_i^2 \qquad \cdots}{A \vdash B.}\; m(i) = m_i^1$$

This interpretation extends to a functor in the obvious way: each move in a TO-play turns into an edge in its interpretation, and a morphism of TO-plays $g \colon s \to t$ is mapped to the morphism of $(A \vdash B)$-trees that maps the edge that corresponds to m to the one that corresponds to $g(m)$. This is indeed a morphism of $(A \vdash B)$-trees because morphisms of TO-plays respect views.

[2] This is not a coproduct in $\mathbb{P}_{A,B}$, but it is a coproduct in the category with the same objects and morphisms, except that morphisms need not be injective.

Lemma 5.3.6. *F is a full embedding.*

Proof. Injectivity on objects is easy to prove, but requires to treat trees formally as pointed presheaves and construct F carefully with this definition, so we skip it. Faithfulness is also easy to prove: if two morphisms of TO-plays $f, g: s \to t$ are different, then there must be an index i such that $f(i) \neq g(i)$, so the morphisms $F(f)$ and $F(g)$ map the edge that corresponds to i to different edges, so they are different. The proof of fullness is mostly straightforward: a morphism of $(A \vdash B)$-trees from $F(s)$ to $F(t)$ is a mapping of the edges of $F(s)$ to those of $F(t)$, which directly corresponds to a mapping of moves in s to those in t. All that is left to show is that this mapping is a morphism of TO-plays, i.e., that it is compatible with the pointers in s and t. This is intuitively the case because morphisms of $(A \vdash B)$-trees map branches to branches, and branches correspond to views in HON games. We show that it is indeed the case by explaining how to recover pointers in s from the structure of $F(s)$, and then show that morphisms of $(A \vdash B)$-trees are necessarily compatible with those pointers. We explain in details how to read pointers from $F(s)$ below. □

First, recovering the moves in s from $F(s)$ is easy: they are all the m's used in any LEFT or RIGHT rule, with the obvious multiplicities. Recovering pointers is a bit trickier and requires to know what an *occurrence* of an arena in a tree is, as well as what it means for a move to *create* such an occurrence, and what it means for a move to be *played on* the occurrence of an arena. Lemmas 5.4.20 and 5.4.24, which are proved independently, then explain how to recover pointers from our structure of plays (these lemmas are expressed in the context of plays as string diagrams, and we translate them here to fit the context of plays as proof trees). They can be stated as follows: each move m is justified by the move that created the arena occurrence m was played on.

Definition 5.3.7. *Let T be an S-tree. The set of* moves *occurring in T is the disjoint union of all the moves of the rules of T, where, in (5.3), there is a single move m occurring in the* LEFT *rule, and the moves that occur in the* RIGHT *rule are all the $m(i)$'s.*

In the LEFT *rule, m* plays on *A and* creates *$A \cdot m$. In the* RIGHT *rule, $m(i)$* plays on *A and* creates *$A \cdot m(i)$. Additionally, if*

$$\frac{\Gamma, A \cdot m(1) \vdash \quad \dots \quad \Gamma, A \cdot m(n) \vdash}{\Gamma \vdash A}$$

is the first rule of T, then $m(i)$ creates all the arenas in $(\Gamma, A \cdot m(i) \vdash)$.

The occurrences *of an arena in an S-tree T is an arena that is either the negative arena in the conclusion of T (if it exists) or an arena created by a rule in T.*

This definition simply means that, if

$$\frac{T_1 \quad \dots \quad T_k}{S'}$$

is a strict sub-tree of an S-tree T, then, for all $i \in |S'|$, all the arenas that appear at the ith position of any sequent in any of the T_j's, is thought of as equal to the arena Γ_i in the sequent at the bottom of this tree. (Note that in the string diagrammatic play that corresponds to T, the elements that correspond to the interpretations of two such arenas are indeed equal).

Example 5.3.8. *There are four occurrences of the empty arena $\varnothing$ in the tree of Figure 5.1, created by t_l, f_l, f_r, and t_r respectively. There are also two occurrences of the boolean arena: the first one exists ($\mathbb{B}_r$) at the beginning, and the second one ($\mathbb{B}_l$) is created by q_r (and shared by all sequents above in the tree).*

In the first tree of Example 5.3.2, there are two occurrences of the A arena, created by both m moves. This is an artefact of playing on the arena pair (A, B) rather than on $A \to B$, but this is more natural in our setting (for example, the notion of interaction sequence*, which is used to study composition of strategies, is the same as that of play, but on particular positions).*

In the tree in Figure 5.1, the q_l move is played on the occurrence of $\mathbb{B}$ called $\mathbb{B}_l$, which is created by q_r, so q_l is justified by q_r. Similarly, f_r is played on the occurrence of $\{\mathsf{t},\mathsf{f}\}$ called $\{\mathsf{t},\mathsf{f}\}_r$, which is also created by q_r, so f_r is also justified by q_r. We thus recover the pointer structure from the P-view tree, i.e., the pointer structure of the play we started from.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r}{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash}\,\mathsf{f}_r \qquad \dfrac{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash \varnothing_r}{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r, \varnothing_l \vdash}\,\mathsf{t}_r}{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash \{\mathsf{t},\mathsf{f}\}_l}\,\mathsf{t}_l, \mathsf{f}_l}{\mathbb{B}_l, \{\mathsf{t},\mathsf{f}\}_r \vdash}\,q_l}{\mathbb{B}_l \vdash \mathbb{B}_r}\,q_r$$

The tree above is the tree from Figure (5.1)*, with colours indicating which move created which arena occurrences. The reader can now readily check that the pointer structure recovered when applying the method described above indeed gives back the pointer structure of the TO-play at the top of Figure* (5.1)*. (However, it is impossible, from the tree structure, to exactly recover the order of moves in the TO-play.)*

Lemma 5.3.9. *F restricts to a functor $F^{\mathbb{V}} : \mathbb{V}_{A,B} \to \mathbb{E}^{\mathbb{V}}(A \vdash B)$.*

Proof. Let us take a TO-view $s = (n, f, \varphi)$, and show that $F(s)$ is a branch. The proof trees of our sequent calculus can only branch with the use of a RIGHT rule. In that case, by the method described above to recover pointers, we get that two Opponent moves are justified by the same Proponent move. But we know by Lemma 2.1.24 that all Opponent moves in s are justified by the preceding move, so there can be no two Opponent moves justified by the same Proponent move. □

Lemma 5.3.10. *$F^{\mathbb{V}}$ is an equivalence of categories.*

Proof. Since i, i_{TO}, and F are fully faithful, $F^{\mathbb{V}}$ is also fully faithful by left cancellation, since fully faithful functors are the right class of a factorisation system (see Example 2.2.87). Now, to show that $F^{\mathbb{V}}$ is essentially surjective on objects, we simply need to build an antecedent through $F^{\mathbb{V}}$ of any view v. The candidate TO-view is given by taking all moves of v from the root to the top (this is unambiguous since v is non-branching) and pointers given by the method described above. All that is left is to verify that the candidate TO-view is indeed a TO-view, which is done by verifying that it is a TO-play, that it has positive length, and that all Opponent moves are justified by the preceding move. The first two points are easy, since the way we have chosen the antecedent may be generalised to any play, and the antecedent verifies all properties of TO-plays, except perhaps for alternation and having positive, even length, which are both trivial in our case because views do not branch and have positive, even depth. The second point is obvious by construction. □

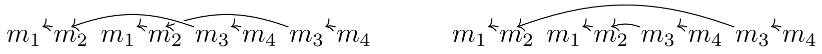
Remark. *The proof above gives some insight on the only fundamental difference between our plays and TO-plays: ours only verify a weak form of alternation, where Opponent may play several moves in a row, but Proponent may only answer once per Opponent move.*

By putting everything together, we get:

Theorem 5.3.11. *The square in Figure 5.1 commutes, F is a full embedding, and $F^{\mathbb{V}}$ is an equivalence of categories.*

5.4 The Level of Plays: Formal Proof

We have just seen a proof that (5.1) commutes, that F is a full embedding, and $F^{\mathbb{V}}$ an equivalence of categories. However, this proof is not very satisfactory. The whole presentation is actually a bit sloppy because the treatment of proof trees is. Indeed, if we define trees, as is usually the case in computer science, as functions from a skeleton to some set of labels, where the skeleton is a set of finite sequences of natural numbers that is closed under prefix (if $(u_1, \ldots, u_{n+1})$ is in a skeleton, then so is $(u_1, \ldots, u_n)$) and predecessor (if $(u_1, \ldots, u_n + 1)$ is in a skeleton, then so is $(u_1, \ldots, u_n)$), then the functor from $\mathbb{P}_{A,B}$ to $\mathbb{T}(A \vdash B)$ is not a full embedding, but only fully faithful. This does not make our final result on categories of strategies any weaker (we still get equivalent categories of strategies), but the link between our plays and Tsukada and Ong's become a bit weaker. As a counterexample, consider the two TO-plays on the arena pair (A, B) below, where all moves are in B, and m_1 is the only initial move in B.


m_1 m_2 m_1 m_2 m_3 m_4 m_3 m_4
m_1 m_2 m_1 m_2 m_3 m_4 m_3 m_4

They are both interpreted as the $(A \vdash B)$-tree

$$\dfrac{\dfrac{\dfrac{\dfrac{A, B_{/m_1}, B_{/m_3} \vdash B_{/m_4}}{A, B_{/m_1}, B_{/m_3} \vdash}}{A, B_{/m_1} \vdash B_{/m_2}}}{A, B_{/m_1} \vdash} \qquad \dfrac{\dfrac{\dfrac{A, B_{/m_1}, B_{/m_3} \vdash B_{/m_4}}{A, B_{/m_1}, B_{/m_3} \vdash}}{A, B_{/m_1} \vdash B_{/m_2}}}{A, B_{/m_1} \vdash}}{A \vdash B.}$$

However, if $(A \vdash B)$-trees are treated as pointed presheaves, then we can get the desired full embedding (in the case of the example shown above, the trees we get are isomorphic, but not equal). The problem is that this requires a careful definition of the functor $F : \mathbb{P}_{A,B} \to \mathbb{T}(A \vdash B)$, and that it is hard to grasp any intuition from the definitions. The problem is not that we only get a fully faithful morphism, but that, no matter which definition we choose for trees, the formal proof is not any easier:

- treating $(A \vdash B)$-trees as labelled skeletons requires a careful management of indices, which is exactly what we do in the formal proofs, but directly between TO-plays and plays,
- while treating $(A \vdash B)$-trees as pointed presheaves does not make the proof any simpler than ignoring $(A \vdash B)$-trees altogether, since plays themselves can be seen as pointed presheaves by Theorem 5.2.15.

We thus rectify the sloppiness of the previous section by formally defining the full embedding $F \colon \mathbb{P}_{A,B} \to \mathbb{E}(A \vdash B)$ and proving that it satisfies all the desired properties. We define it directly, without using $\mathbb{T}(A \vdash B)$, as a formal treatment of trees does not make the definitions any simpler.

In Section 5.4.1, we actually construct our functor F from TO-plays to plays, which is proved in Section 5.4.2 to be fully faithful. We then show in Section 5.4.3 that F restricts to a functor $F^{\mathbb{V}}$ from TO-views to views, which is an equivalence of categories.

5.4.1 Constructing the Functor

Required: 5.2.3.
Recommended: 5.3.3.

In this section, we define the functor $F \colon \mathbb{P}_{A,B} \to \mathbb{E}(A \vdash B)$. By Theorem 5.2.15, constructing F reduces to defining a fully-faithful functor $F \colon \mathbb{P}_{A,B} \to \mathbb{E}'(A \vdash B)$ restricts to views, as in:

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r, hook, "\mathsf{i}_{TO}"] \arrow[d, "F^{\mathbb{V}}"'] & \mathbb{P}_{A,B} \arrow[d, "F"] \\
(\mathbb{E}^{\mathbb{V}})'(A \vdash B) \arrow[r, hook, "\mathsf{i}"'] & \mathbb{E}'(A \vdash B).
\end{tikzcd} \tag{5.7}$$

We first construct F on objects, which requires some preparatory notation on arenas and preplays.

Notation 5.4.1. *We define $(A,B)_{/m} = A_{/m}$ when m is in A and $(A,B)_{/m} = B_{/m}$ when m is in B.*

We immediately notice:

Lemma 5.4.2. *For all m, m' in $M_A + M_B$, if $m \vdash_{A \multimap B} m'$ and $\star \nvdash_A m'$, then $m' \in \sqrt{(A,B)_{/m}}$ and $(A,B)_{/m'} = (A,B)_{/m} \cdot m'$.*

Proof. Trivial. □

Notation 5.4.3. *In any preplay $s = (n, f, \varphi)$ and $i \in n$, let $K_s(i)$ denote the length of $\lceil s \rceil_i$, and let $\lceil s \rceil_i = (I_1^{s,i}, \ldots, I_{K_s(i)}^{s,i})$ (where $\lceil s \rceil_i$ is the sequence of all indices of moves in the view of s at index i, so $I_j^{s,i}$ are numbers). Furthermore, let $I_0^{s,i} = 0$ for all $i \in n$.*

By recasting the definition of views in HON games in terms of $K_s(i)$ and $I_j^{s,i}$, we get a few useful relations. First, we can follow the $K_s(-)$'s through a view. Indeed, for all $i \in n$:

- if i is odd, then $K_s(i) = 1 + K_s(\varphi(i))$ and for all $j < K_s(i)$, $I_j^{s,i} = I_j^{s,\varphi(i)}$, with the convention that $K_s(0) = 0$;
- if i is even, then $K_s(i) = 1 + K_s(i-1)$ and for all $j < K_s(i)$, $I_j^{s,i} = I_j^{s,i-1}$.

Furthermore, the definition of views reflects into the following relations:

- $I_{K_s(i)}^{s,i} = i$;
- for all odd $k \in K_s(i)$, $I_{k-1}^{s,i} = \varphi(I_k^{s,i})$;
- for all even $k \in K_s(i)$, $I_{k-1}^{s,i} = I_k^{s,i} - 1$.

We often omit both superscripts when clear from context. For all $j \in K_s(i)$, j and I_j have the same parity. This in particular gives φ in terms of I, for odd (i.e., Opponent) moves. In the case of even (or Proponent) moves, by P-visibility, when j is even, we have $\varphi(j) \in \lceil s \rceil_i$ and so there exists a unique $l \in K_s(i)$ such that $\varphi(j) = I_l$. By alternation, $\varphi(j)$ is odd and so l also is. Thus, there exists a unique $L_s(j) \in K_s(i)/2$ such that $l = 2L_s(j) - 1$. In summary, we have:

Lemma 5.4.4. *For all $i \in n$ and $j \in \lceil s \rceil_i$, letting $k \in K_s(i)$ be such that $I_k^{s,i} = j$, we have*

- $\varphi(j) = I_{2L_s(j)-1}$ *if j is even and*
- $\varphi(j) = \varphi(I_k^{s,i}) = I_{k-1}^{s,i}$ *if j is odd.*

Lemma 5.4.5. *For all preplays s, $i \in |s|$, and $j \in \lceil s \rceil_i$, $K_s(j) \leq K_s(i)$ and for all $k \in K_s(j)$, $I_k^{s,j} = I_k^{s,i}$.*

Proof. By induction on i. If i is odd and $\varphi(i) = 0$, then the result is obvious. If i is odd and $\varphi(i) > 0$, then either $j = i$, in which case the result is obvious, or $j \in \lceil s \rceil_{\varphi(i)}$, in which case $K_s(j) \leq K_s(\varphi(i)) = K_s(i) - 1$ by induction hypothesis, and for all $k \in K_s(j)$, $I_k^{s,j} = I_k^{s,\varphi(i)} = I_k^{s,i}$ by induction hypothesis and the fact that $k < K_s(i)$. If i is even, the proof follows the same pattern. □

Furthermore, the sequence of all I_{2l-1} for $l \in \frac{K_s(i)+1}{2}$ (where the division is understood in the integer setting and thus in particular equals $\frac{K_s(i)}{2}$ when i is even) is relevant in our context. It consists of all odd indices in $\lceil s \rceil_i$. Let us provide some notation for this:

Notation 5.4.6. *For all maps $x : n \to X$ to some set X, let us denote by $[x(i)]_{i \in n}$ the sequence $(x(1), \ldots, x(n))$.*

So, e.g., the above subsequence of I is denoted by $[I_{2l-1}]_{l \in \frac{K_s(i)+1}{2}}$. Using this notation, we may explicitly characterise the sequents associated to each stage of s. As we will see below, for each move $i \in n$, $S^{s,i+1}$ will be the sequent of the player "created by the ith move" in $F(s)$.

Notation 5.4.7. *For any justified sequence $s = (n, f, \varphi)$, we define $(A,B)^s_{/i} = (A,B)_{/f(i)}$. We extend this by convention to $(A,B)^s_{/0} = B$.*

Definition 5.4.8. *For any $i \in n \uplus \{0\}$, let $S^{s,i+1}$ denote the sequent defined by*

- $(A, [(A,B)^s_{/I^{s,i}_{2l-1}}]_{l \in \frac{K_s(i)+1}{2}} \vdash)$ *if i is odd and*
- $(A, [(A,B)^s_{/I^{s,i}_{2l-1}}]_{l \in \frac{K_s(i)}{2}} \vdash (A,B)^s_{/I^{s,i}_{K_s(i)}})$ *if i is even.*

In particular, when $i = 0$, the definition yields $S^{s,1} = (A \vdash B)$.

First, let us observe:

Lemma 5.4.9. *For all $i \in |s|$, if $f(i) \notin \sqrt{A}$, then $f(i) \in \sqrt{(A,B)^s_{/\varphi(i)}}$ and $(A,B)^s_{/i} = (A,B)^s_{/\varphi(i)} \cdot f(i)$.*

Proof. If $\varphi(i) = 0$, then $f(i) \in \sqrt{B}$, so the formula obviously holds because $(A,B)^s_{/0} = B$. Otherwise, the result is a direct application of Lemma 5.4.2, using the fact that $f(\varphi(i)) \vdash_{A,B} f(i)$. □

Which entails:

Corollary 5.4.10. *The arenas $(A,B)^s_{/I_1}, \ldots, (A,B)^s_{/I_{K_s(i)}}$ are related by*

- $(A,B)^s_{/I_k} = (A,B)^s_{/I_{k-1}} \cdot f(I_k)$ *when k is odd,*
- $(A,B)^s_{/I_k} = (A,B)^s_{/I_{2L_s(I_k)-1}} \cdot f(I_k)$ *when k is even and $f(I_k) \notin \sqrt{A}$, and*
- $(A,B)^s_{/I_k} = A \cdot f(I_k)$ *when k is even and $f(I_k) \in \sqrt{A}$,*

for all $k \in K_s(i)$.

Proof. By Lemmas 5.4.4 and 5.4.9 for the first two points. For the third point, we have $(A,B)^s_{/I_k} = (A,B)_{/f(I_k)} = A_{/f(I_k)} = A \cdot f(I_k)$. □

We now show a few useful lemmas about TO-plays and morphisms of such. The first one simply states that morphisms of TO-preplays map views to views:

Lemma 5.4.11. *If $g\colon s \to s'$ is a morphism of TO-preplays, then for all $i \in |s| \uplus \{0\}$:*

- $K_{s'}(g(i)) = K_s(i)$,
- *for all $j \in K_s(i)$, $I^{s',g(i)}_j = g(I^{s,i}_j)$.*

Proof. Let us start with the first point, by induction on i. For the base case, we have $K_{s'}(g(0)) = K_{s'}(0) = 0 = K_s(0)$ by definition. Now, for the induction step, consider $i > 0$. If i is odd:

$$\begin{aligned}
K_{s'}(g(i)) &= 1 + K_{s'}(\varphi'(g(i))) && \text{because } g(i) \text{ is odd} \\
&= 1 + K_{s'}(g(\varphi(i))) && \text{because } \varphi' g = g\varphi \\
&= 1 + K_s(\varphi(i)) && \text{by induction hypothesis} \\
&= K_s(i) && \text{because } i \text{ is odd,}
\end{aligned}$$

and similarly when i is even.

For the second point, we again proceed by induction on i. The base case trivially holds because $K_s(i) = 0$. For the induction step, consider any $i > 0$. If i is odd, then for all $j < K_s(i)$:

$$\begin{aligned}
I_j^{s',g(i)} &= I_j^{s',\varphi'(g(i))} && \text{because } g(i) \text{ is odd and } j < K_s(i) = K_{s'}(g(i)) \\
&= I_j^{s',g(\varphi(i))} && \text{because } \varphi' g = g\varphi \\
&= g(I_j^{s,\varphi(i)}) && \text{by induction hypothesis} \\
&= g(I_j^{s,i}) && \text{because } i \text{ is odd and } j < K_s(i),
\end{aligned}$$

and similarly if i is even. Finally, when $j = K_s(i)$, we have:

$$\begin{aligned}
I^{s',g(i)}_{K_s(i)} &= I^{s',g(i)}_{K_{s'}(g(i))} && \text{by the first point} \\
&= g(i) \\
&= g(I^{s,i}_{K_s(i)})
\end{aligned}$$

as desired. □

Lemma 5.4.12. *If $g: s \to s'$ is a morphism of TO-preplays, then for all $i \in \{0\} \uplus |s|$, $(A,B)^{s'}_{/g(i)} = (A,B)^{s}_{/i}$.*

Proof. We have $(A,B)^{s'}_{/g(i)} = (A,B)_{/f'(g(i))} = (A,B)_{/f(i)} = (A,B)^{s}_{/i}$. □

Lemma 5.4.13. *If $g: s \to s'$ is a morphism of TO-preplays, then for all $i \in \{0\} \uplus |s|$, $S^{s,i+1} = S^{s',g(i)+1}$.*

Proof. When i is odd, then $g(i)$ is also odd, so

$$S^{s',g(i)+1} = (A, [(A,B)^{s'}_{/I^{s',g(i)}_{2l-1}}]_{l \in \frac{K_{s'}(g(i))+1}{2}} \vdash).$$

By Lemma 5.4.11, we know that $K_{s'}(g(i)) = K_s(i)$ and that for all $l \in \frac{K_{s'}(g(i))+1}{2}$, $I^{s',g(i)}_{2l-1} = g(I^{s,i}_{2l-1})$, which directly implies the result by Lemma 5.4.12. The proof is similar when i is even. □

Lemma 5.4.14. *If $g: s \to s'$ is a morphism of TO-preplays, then for any even $i \in |s|$, we have $L_{s'}(g(i)) = L_s(i)$.*

Proof. Since $\varphi'(g(i)) = g(\varphi(i))$, we know that

$$I^{s',g(i)}_{2L_{s'}(g(i))-1} = \varphi'(g(i)) = g(\varphi(i)) = g(I^{s,i}_{2L_s(i)-1}) = I^{s',g(i)}_{2L_s(i)-1}$$

by Lemma 5.4.11, which entails the desired result, since $k \mapsto I^{s',g(i)}_k$ is monic. □

Returning to the construction of F on objects, in fact, we construct it on all preplays in $\mathbb{PP}_{A,B}$, by induction on their length, and eventually restrict to plays. In order for our induction step to make sense, we should make explicit a few invariants that our assignment will satisfy.

Notation 5.4.15. *In the following, we will denote $x \cdot f$ by $x \odot f$ when x is a player, and use $x \cdot f$ only when x is a move. (This is useful because all channels, players, and moves will be natural numbers, so this notation should spare the reader a few headaches.)*

For any preplay $s = (n, f, \varphi)$, the associated morphism $F(s) = ((A \vdash B) \xrightarrow{t} U)$ in $\mathbb{E}'(A \vdash B)$ will satisfy:

(P1) The sets $U(A)$ for all arenas A are pairwise disjoint, and $\biguplus_{A \in \mathbb{A}} U(A) = n+2$; this induces a map $\mathsf{a}: n+2 \to \mathbb{A}$, where $\mathbb{A}$ denotes the set of all sub-arenas of A or B, such that for all $i \in n+2$, $i \in U(\mathsf{a}(i))$;

(P2) The sets $U(S)$ for all sequents S are pairwise disjoint, and $\biguplus_{S \in \mathbb{S}} U(S) = n+1$; this induces a map $\mathsf{s}: n+1 \to \mathbb{S}$, where $\mathbb{S}$ denotes the set of all sequents of sub-arenas of A or B, such that for all $i \in n+1$, $i \in U(\mathsf{s}(i))$;

(P3) The sets $U(\mu)$ for all moves $\mu \in \mathrm{ob}(\mathbb{L}_{|2})$ are pairwise disjoint, and we have $\biguplus_{\mu \in \mathbb{L}_{|2}} U(\mu) = n$; this induces a map $\mathsf{m}: n \to \mathrm{ob}(\mathbb{L}_{|2})$ such that for all $i \in n$, $i \in U(\mathsf{m}(i))$;

(P4) for all $i \in n$, $i \cdot s = i+1$ (move i creates player $i+1$);

(P5) the player $n+1$ is final in U;

(P6) for all $i \in n+1$, $\mathsf{s}(i) = S^{s,i}$.

When $n = 0$, the preplay s is mapped by definition to a morphism that is isomorphic to the identity on $(A \vdash B)$ and that verifies the conditions above. There are two such morphisms, and we (arbitrarily) choose $(A \vdash B) \Rrightarrow U$ such that

$$U(A \vdash B) = \{1\}, \qquad 1 \odot s_1 = 1, \qquad 1 \odot t = 2. \tag{5.8}$$

For the induction step, consider any $s = (n+1, f, \varphi)$ and assume that $(A \vdash B) \xrightarrow{t'} U'$ satisfies (P1)–(P6) for the immediate prefix s' of s.

We define $F(s) = ((A \vdash B) \xrightarrow{t} U)$ by a specific choice of composite $U' \bullet M$, for some move $M: \uparrow U \mathrel{-\!\!\bullet\!\!\!\to} \uparrow U'$, itself constructed by choosing some seed $Y_0 \xrightarrow{s_0} \mu \xleftarrow{t_0} X_0$ with representable X_0 and final player $x \in U'(X_0)$, and taking the pushout

$$\begin{tikzcd}
X_0 \arrow[r, "t_0"] \arrow[d, "\ulcorner x \urcorner"'] & \mu \arrow[d] \arrow[dl, phantom, "\ulcorner" very near end] \\
\uparrow U' \arrow[r] & M,
\end{tikzcd}$$

which ensures that $F(s)$ is in $\mathbb{E}'(A \vdash B)$. The map t_0 being injective, so is the induced map $U' \to U$. We choose to make it an inclusion, which makes it entirely determined by the choice of "created" elements, i.e., of images of elements in $\mu \smallsetminus X_0$. Because $\mu \in \mathrm{ob}(\mathbb{L}_{|2})$, we know that in all cases there is exactly one created channel, one created player, and one new move. Since the channels, players and moves of U' respectively are the elements of $n+2, n+1$ and n, we will systematically pick the created elements to be respectively $n+3, n+2$ and $n+1$. Thus, we only need to choose x final in U' and μ and show that (P6) is preserved. Let us proceed by case analysis:

- If $n+1$ is odd and $\varphi(n+1) = 0$, then $f(n+1) \in \sqrt{B}$ so we may pick $\mu = \Lambda_{(A \vdash B), f(n+1)}$, which leaves us with the task of picking some player in $U'(A \vdash B)$. But by (P6), $1 \in U'(S^{s,1}) = U'(A \vdash B)$: we simply pick 1. To show that (P6) is preserved, we compute $S^{s,n+2}$ to check that it is equal to $(A, B{\cdot}f(n{+}1) \vdash)$, which comes from the facts that $K_s(n{+}1) = 1{+}K_s(\varphi(n{+}1)) = 1{+}0 = 1$ and $(A,B)^s_{/I_1^{s,n+1}} = (A,B)^s_{/I^{s,n+1}_{K_s(n+1)}} = (A,B)^s_{/n+1} = B{\cdot}f(n{+}1)$.

- If $n+1$ is odd and $\varphi(n+1) \neq 0$, then let $i = \varphi(n+1)$. Thus, we have $f(i) \vdash_{A \multimap B} f(n+1)$, so by Lemma 5.4.2 we have $f(n+1) \in \sqrt{(A,B)^s_{/i}}$ (because $n{+}1$ is odd, so $f(n{+}1)$ is not initial in A). Furthermore, by (P4) and (P6), $(i \cdot s) = (i+1)$ is in $U(S^{s',i+1})$, which, because i is even by alternation, is equally

$$U(A, [(A,B)^{s'}_{/I^{s',i}_{2l-1}}]_{l \in \frac{K_{s'}(i)}{2}} \vdash (A,B)^{s'}_{/I^{s',i}_{K_{s'}(i)}}).$$

  But $I^{s',i}_{K_{s'}(i)} = i$, so $(A,B)^{s'}_{/I^{s',i}_{K_{s'}(i)}} = (A,B)^{s'}_{/i} = (A,B)^s_{/i}$: we pick $x = (i+1)$ and $\mu = \Lambda_{S^{s',i+1},f(n+1)}$. We thus have that $n+2$ is a player over

$$(A, [(A,B)^{s'}_{/I^{s',i}_{2l-1}}]_{l \in \frac{K_{s'}(i)}{2}}, (A,B)^{s'}_{/I^{s',i}_{K_{s'}(i)}} \cdot f(n+1) \vdash),$$

  which is $S^{s,n+2}$ as desired, because $K_{s'}(i) = K_s(i) = K_s(n+1) - 1$ (for the number of arenas), $(A,B)^{s'}_{/I^{s',i}_{2l-1}} = (A,B)_{/I^{s,i}_{2l-1}} = (A,B)_{/I^{s,n+1}_{2l-1}}$ (to show that the bracketed arenas are equal to those of $S^{s,n+2}$), and $(A,B)^{s'}_{/I^{s',i}_{K_{s'}(i)}} \cdot f(n{+}1) = (A,B)_{/I^{s,i}_{K_s(i)}} \cdot f(n{+}1) = (A,B)_{/I^{s,n+1}_{K_s(n+1)-1}} \cdot f(I^{s,n+1}_{K_s(n+1)}) = (A,B)_{/I^{s,n+1}_{K_s(n+1)}}$ by Corollary 5.4.10 (to show that the last arena is equal to that of $S^{s,n+2}$).

- If $n+1$ is even, then we pick $x = (n \cdot s) = n+1$ (by (P4)). Moreover, $n+1$ is in $U(S^{s',n+1})$ by (P6), which, because n is odd, is equally

$$U(A, [(A,B)^{s'}_{/I^{s',n}_{2l-1}}]_{l \in \frac{K_{s'}(n)+1}{2}} \vdash).$$

  If now $f(n{+}1) \in \sqrt{A}$, then we pick $\mu = @_{S^{s',n+1},1,f(n+1)}$. Otherwise, taking $k = n+1$ in Corollary 5.4.10 yields

$$(A,B)^s_{/n+1} = (A,B)^s_{/I^{s,n}_{2L_s(n+1)-1}} \cdot f(n+1) = (A,B)^{s'}_{/I^{s,n}_{2L_s(n+1)-1}} \cdot f(n+1).$$

  But since $n+1$ is even and $2L_s(n+1) - 1 < K_s(n+1)$, $I^{s,n}_{2L_s(n+1)-1} = I^{s',n}_{2L_s(n+1)-1}$, so in particular $f(n+1) \in \sqrt{(A,B)^{s'}_{/I^{s',n}_{2L_s(n+1)-1}}}$ and, in view of the form of $S^{s',n+1}$, we may pick $\mu = @_{S^{s',n+1},1+L_s(n+1),f(n+1)}$. The fact that $n{+}2$ is in $U(S^{s,n+2})$ follows by a computation similar to the one done in the previous case.

The chosen player is final in U' by definition when $n+1$ is odd, and by (P5) otherwise.

We now want to show that F as defined here extends to a functor. We will define the image of a morphism of TO-plays below, but we first need a few preparatory lemmas.

We first state and prove a few lemmas about the presheaf U obtained by applying the construction above.

Lemma 5.4.16. *For all preplays s, if $F(s) = ((A \vdash B) \to U)$, then for all $i \in |s|$:*

- *$i \in U(\Lambda_{S^{s,\varphi(i)+1},f(i)})$ if i is odd,*
- *$i \in U(@_{S^{s,i},1,f(i)})$ if i is even and $f(i) \in \sqrt{A}$,*
- *$i \in U(@_{S^{s,i},1+L_s(i),f(i)})$ if i is even and $f(i) \notin \sqrt{A}$.*

Proof. By induction on s:

- if s is empty, then the result is obvious,
- otherwise, there are two possible cases. Either $i < |s|$, in which case, if we denote by s' the immediate prefix of s, and $F(s') = ((A \vdash B) \to U')$, we know by induction hypothesis that i belongs to $U'(\mu) \subseteq U(\mu)$ for the desired μ (because $S^{s,i} = S^{s',i}$ and $L_s(i) = L_{s'}(i)$). Or $i = |s|$, in which case the result holds by construction. □

We can now state and prove the lemma we are interested in:

Lemma 5.4.17. *If $g: s \to s'$ is a morphism of TO-plays, $F(s) = ((A \vdash B) \to U)$, and $F(s') = ((A \vdash B) \to U')$, then:*

- *for all $\mu \in \mathbb{L}_{|2}$, if $i \in U(\mu)$, then $g(i) \in U'(\mu)$,*
- *for all $S \in \mathbb{L}_{|1}$, if $i + 1 \in U(S)$, then $g(i) + 1 \in U'(S)$,*
- *for all $C \in \mathbb{L}_{|0}$, if $i + 2 \in U(C)$, then $g(i) + 2 \in U'(C)$.*

Proof. For the first point, let us assume that i is odd. We know that if i is in $U(\mu)$, then by Lemma 5.4.16 we have $\mu = \Lambda_{S^{s,\varphi(i)+1},f(i)}$. But then $S^{s,\varphi(i)+1} = S^{s',g(\varphi(i))+1} = S^{s',\varphi'(g(i))+1}$ by Lemma 5.4.13 and the fact that $g\varphi = \varphi' g$, so $\mu = \Lambda_{S^{s',\varphi'(g(i))+1},f'(g(i))}$ (since $f'(g(i)) = f(i)$), so by Lemma 5.4.16 we have $g(i) \in U'(\mu)$. The proof is similar in the other cases (note that, in particular, the proof uses Lemma 5.4.14 when i is even and $f(i) \notin \sqrt{A}$).

The proofs of the other two points follow a similar pattern (alternatively, one may notice that, for all k in $|s|$, $\mathsf{s}(k+1)$ and $\mathsf{a}(k+2)$ are determined by $\mathsf{m}(k)$ by construction). □

We will use the following notation to define $F(g)$:

Definition 5.4.18. *If f is a function from n to m, we define $\tilde{f}: n+1 \to m+1$ by $\tilde{f}(1) = 1$ and $\tilde{f}(i+1) = f(i) + 1$ for all $i \in n$.*

The following lemma will be useful later:

Lemma 5.4.19. *If $g: (n, f, \varphi) \to (n', f', \varphi')$ is a morphism of TO-plays, then $\tilde{g}(2i) = g(2i)$.*

Proof. We have $\tilde{g}(2i) = \tilde{g}(2i-1+1) = g(2i-1)+1 = g(2i)$ because g is a morphism of TO-plays. □

Now, if $g: s \to s'$ is a morphism of TO-plays, $F(s) = ((A \vdash B) \to U)$, and $F(s') = (X \to U')$, we define $k = F(g): U \to U'$ as:

- $k_\mu(x) = g(x)$ for all $\mu \in \mathbb{L}_{|2}$ and $x \in U(\mu)$,
- $k_S(x) = \tilde{g}(x)$ for all $S \in \mathbb{L}_{|1}$ and $x \in U(S)$,
- $k_A(x) = \tilde{\tilde{g}}(x)$ for all $A \in \mathbb{L}_{|0}$ and $x \in U(A)$.

This is indeed well-defined by Lemma 5.4.17 and (5.8). We need to prove two other lemmas in order to show that k is natural.

Lemma 5.4.20. *For all preplays s and $i \in |s|$, if $F(s) = ((A \vdash B) \to U)$, then:*

- *$i \cdot t = i$ if i is even,*
- *$i \cdot t = \varphi(i) + 1$ if i is odd.*

Proof. By induction on s and case distinction on i, where the only interesting case is when $i = |s|$, in which case both points hold by construction. □

By (P4), the player created by each move i is $i+1$. The next lemma describes how its associated sequent, $S^{s,i+1}$ (Definition 5.4.8), connects to its environment. Intuitively, this is quite simple: all non-new connections are as in $S^{s,i}$; and the new formula connects to the created channel, $i+2$. The technical statement is obfuscated by index handling, so let us explain this a bit. By construction, if i is odd, $S^{s,i+1}$ is positive and has $1 + \frac{K_s(i)+1}{2}$ hypotheses. If i is even, $S^{s,i+1}$ is negative and has $1 + \frac{K_s(i)}{2}$ hypotheses. In both cases, the first hypothesis is A, and the others have been created by previous moves. In fact, when i is even, so is $K_s(i)$, so $1 + \frac{K_s(i)}{2} = 1 + \frac{K_s(i)+1}{2}$. So for all $i \in n$, $S^{s,i+1}$ has $1 + \frac{K_s(i)+1}{2}$ hypotheses.

Lemma 5.4.21. *For all preplays s, $i \in |s|$, if $F(s) = ((A \vdash B) \to U)$, then, recalling Definition 5.4.8 and the fact that $i \cdot s = i + 1$, we have:*

- *if i is odd:*
  - *$(i+1) \odot s_j = (\varphi(i)+1) \odot s_j$ for all $j \in \frac{K_s(i)+1}{2}$ and*
  - *$(i+1) \odot s_{\frac{K_s(i)+1}{2}+1} = i+2$;*
- *if i is even:*
  - *$(i+1) \odot s_j = i \odot s_j$ for all $j \in \frac{K_s(i)}{2} + 1$,*
  - *$(i+1) \odot t = i + 2$.*

Proof. By induction on s and case distinction on i, where the only interesting case is when $i = |s|$, in which case all points hold by construction. □

We can now prove that k defined above is a natural transformation, which amounts to showing that all diagrams of the form

$$\begin{tikzcd} U(\mu) \arrow[r, "k_\mu"] \arrow[d, "U(\alpha)"'] & U'(\mu) \arrow[d, "U'(\alpha)"] \\ U(S) \arrow[r, "k_S"'] & U'(S) \end{tikzcd} \quad \text{or} \quad \begin{tikzcd} U(S) \arrow[r, "k_S"] \arrow[d, "U(\beta)"'] & U'(S) \arrow[d, "U'(\beta)"] \\ U(C) \arrow[r, "k_C"'] & U'(C) \end{tikzcd} \qquad (5.9)$$

commute, for all $\alpha{:}\, S \to \mu$ and $\beta{:}\, C \to S$. The left-hand side diagram can now be shown to commute using Lemma 5.4.20, and the right-hand side one using Lemma 5.4.21 within an induction. For example, the following computation shows that the right-hand side diagram commutes for $i > 1$ odd and $\beta = s_j$ for $j < \frac{K_s(i)+1}{2} + 1$ ($i = 1$ can easily be verified by hand):

$$\begin{aligned}
\tilde{\tilde{g}}(i \odot s_j) &= \tilde{\tilde{g}}((\varphi(i-1)+1) \odot s_j) && \text{by Lemma 5.4.21} \\
&= \tilde{g}(\varphi(i-1)+1) \odot s_j && \text{by induction hypothesis} \\
&= (g(\varphi(i-1))+1) \odot s_j && \\
&= (\varphi'(g(i-1))+1) \odot s_j && \text{because } g\varphi = \varphi' g \\
&= (g(i-1)+1) \odot s_j && \text{by Lemma 5.4.21} \\
&= \tilde{g}(i) \odot s_j . &&
\end{aligned}$$

The other points either follow a similar pattern or can be proved directly by Lemma 5.4.20 or 5.4.21.

When restricted to $(A \vdash B)$, k turns into $id_{(A \vdash B)}$, as desired. Finally, k is 1D-injective by injectivity of g, which ends the definition of F on morphisms.

It remains to prove functoriality of F, which follows directly from functoriality of $\tilde{-}$.

5.4.2 Full Faithfulness

Required: 5.4.1.
Recommended: ∅.

This section is devoted to proving:

Theorem 5.4.22. *The functor $F{:}\, \mathbb{P}_{A,B} \to \mathbb{E}'(A \vdash B)$ is a full embedding.*

We start by giving two lemmas that express φ in function of $F(s)$, making injectivity on objects of F obvious. Remember that, for all i in n, $S^{s,i}$ has $\frac{K_s(i)+1}{2} + 1$ hypotheses.

Lemma 5.4.23. *For all preplays s, $i \in |s|$, and $j \in \lceil s \rceil_i$, and $k \in \frac{K_s(j)+1}{2} + 1$: $(j+1) \odot s_k = (i+1) \odot s_k$.*

Proof. By induction on $i - j$ and Lemma 5.4.21. □

The next lemma expresses the justifier $\varphi(i)$ of i in terms of $F(s)$, using the fact that moves and created channels are in bijection, each move i creating channel $i+2$. The idea here is that if some positive move i is played on a sequent $(A_1, \ldots, A_N \vdash)$, say on A_k, then $\varphi(i)$ should be the move that has created A_k. But, i being positive, the involved player is just i itself, and the corresponding channel is $i \odot s_k$, which has been created by move $i \odot s_k - 2$. There is one exception though: when $k = 1$, the move is played on A, so by definition of $A \multimap B$, its justifier is the first move played on B in the corresponding view. But this is precisely the move having created $i \odot s_2$, i.e., $i \odot s_2 - 2$.

Lemma 5.4.24. *For all preplays s, if $F(s) = ((A \vdash B) \to U)$ and $i \in U(@_{S,k,m})$, then:*

- $\varphi(i) = i \odot s_k - 2$ *if* $k > 1$,
- $\varphi(i) = i \odot s_2 - 2$ *if* $k = 1$.

Proof. Let us assume $k > 1$ (the other case is similar). By Lemma 5.4.16, $k = 1 + L_s(i)$. Now, on the one hand $\varphi(i) = I^{s,i}_{2L_s(i)-1}$, and on the other hand $\varphi(i) = I^{s,\varphi(i)}_{K_s(\varphi(i))} = I^{s,i}_{K_s(\varphi(i))}$, so $2L_s(i) - 1 = K_s(\varphi(i))$, since $k \mapsto I^{s,i}_k$ is monic. Therefore, we have $k = \frac{K_s(\varphi(i))+1}{2} + 1$, so since $\varphi(i) \in \lceil s \rceil_{i-1}$, Lemma 5.4.23 entails that $(\varphi(i) + 1) \odot s_k = i \odot s_k$. But, by Lemma 5.4.21, since $k = \frac{K_s(\varphi(i))+1}{2} + 1$, we also have that $(\varphi(i) + 1) \odot s_k = \varphi(i) + 2$. We thus derive $\varphi(i) + 2 = i \odot s_k$, hence the result. □

Now, faithfulness is easy: if g and g' are two morphisms $s \to s'$, then, letting $k = F(g)$ and $k' = F(g')$, we know by (P3) that the $U(\mu)$'s form a partition of n, and by definition of k and k' that $k_\mu(x) = g(x)$ and $k'_\mu(x) = g'(x)$ for all $x \in F(s)(\mu)$ and $\mu \in \mathbb{L}_{|2}$. Therefore, if $k_\mu = k'_\mu$ for all $\mu \in \mathbb{L}_{|2}$, then $g = g'$.

Proving fullness is a bit more involved. Let us start with a lemma asserting that any natural transformation $k: F(s) \to F(s')$ is layered just as any $F(g)$. In order to formalise this, let us state:

Definition 5.4.25. *For any TO-play s and $i \in \{0, 1, 2\}$, we define $F(s)_{|i} = \biguplus_{c \in \mathbb{L}_{|i}} F(s)(c)$.*

For any $k: F(s) \to F(s')$ in $\mathbb{E}'(A \vdash B)$ and $i \in \{0, 1, 2\}$, let

$$k_{|i}: F(s)_{|i} \to F(s')_{|i}$$

denote the restrictions of k to each given domain and codomain.

When $U = F(s)$ for some $s = (n, f, \varphi)$, we know by construction that $F(s)_{|2} = n$, $F(s)_{|1} = n + 1$, and $F(s)_{|0} = n + 2$. This allows us to state:

Lemma 5.4.26. *For any $k: F(s) \to F(s')$ in $\mathbb{E}'(A \vdash B)$,*

$$k_{|1} = \widetilde{k_{|2}} \qquad\qquad \textit{and} \qquad\qquad k_{|0} = \widetilde{\widetilde{k_{|2}}}.$$

Proof. Because $\mathbb{E}'(A \vdash B)$ is a subcategory of the coslice $(A \vdash B)/\widehat{\mathbb{L}}$, we know that $k_{|1}(1) = 1$, $k_{|0}(1) = 1$, and $k_{|0}(2) = 2$. It thus remains to prove that $k_{|1}(i + 1) = k_{|2}(i) + 1$ and $k_{|0}(i + 2) = k_{|1}(i + 1) + 1$, for all $i \in n$.

We have

$$\begin{aligned} k_{|1}(i + 1) &= k_{|1}(i \cdot s) \\ &= k_{|2}(i) \cdot s \qquad \text{(by naturality of } k\text{)} \\ &= k_{|2}(i) + 1 \end{aligned}$$

for the first point. We treat the second point by case analysis on the parity of i, using naturality and Lemma 5.4.21:

$$\begin{aligned} k_{|0}(2i + 1) &= k_{|0}((2i) \odot s_N) \\ &= k_{|1}(2i) \odot s_N \\ &= k_{|1}(2i) + 1 \end{aligned} \qquad \text{and} \qquad \begin{aligned} k_{|0}(2i + 2) &= k_{|0}((2i + 1) \odot t) \\ &= k_{|1}(2i + 1) \odot t \\ &= k_{|1}(2i + 1) + 1 \end{aligned}$$

where in the odd case $N = \frac{K_s(2i-1)+1}{2} + 1$. □

We are finally able to prove that F is full. Consider any $s = (n, f, \varphi)$ and $s' = (n', f', \varphi')$, with $F(s) = ((A \vdash B) \to U)$ and $F(s') = ((A \vdash B) \to U')$, together with a morphism $k: U \to U'$ in $\mathbb{E}'(A \vdash B)$. We know by (P3) that the $U(\mu)$'s form a partition of n for μ in $\mathbb{L}_{|2}$. We can therefore define the map $g: n \to n'$ by $g = k_{|2}$. Our goal is to show that g is a morphism $s \to s'$ of TO-plays and that $F(g) = k$. But given the first point, the latter follows from $g = k_{|2}$ by definition of F and Lemma 5.4.26.

So let us show that g is a morphism of TO-plays.

- To show that $f'(g(i)) = f(i)$, we notice that $g(i)$ belongs to $U'(\mathsf{m}'(g(i)))$, but also to $U'(\mathsf{m}(i))$, because $g(i) = k_{|2}(i) = k_{\mathsf{m}(i)}(i): U(\mathsf{m}(i)) \to U'(\mathsf{m}(i))$. Since all $U'(\mu)$ are disjoint, for μ in $\mathbb{L}_{|2}$, we have that $\mathsf{m}(i) = \mathsf{m}'(g(i))$. Now, by construction, $\mathsf{m}(i)$ is either $\Lambda_{S,f(i)}$ or $@_{S,k,f(i)}$, and $\mathsf{m}'(g(i))$ either $\Lambda_{S,f'(g(i))}$ or $@_{S,k,f'(g(i))}$, hence the result.

- Furthermore, g is injective by 1D-injectivity of k.

- We also know that k is natural, so we get commuting diagrams as in (5.9) for all $\alpha: S \to \mu$ and $\beta: C \to S$. By taking $\alpha = s$, we get that $g(2i-1) = g(2i) - 1$ and $g(\varphi(2i-1)) = \varphi'(g(2i-1))$ for all $i \in n/2$. The last point we need to show is that $g(\varphi(2i)) = \varphi'(g(2i))$ for all $i \in n/2$. Let us consider the case where $\mathsf{m}(2i) = @_{S,k,m}$ and $k > 1$ (the proof is similar for $k = 1$). By alternation, $\varphi(2i)$ is odd, so $\varphi(2i) \geq 1$, hence $(2i) \odot s_k = \varphi(2i) + 2 \geq 3$, by Lemma 5.4.24, since $k > 1$. Thus, by definition, $\tilde{\tilde{g}}((2i) \odot s_k) = g((2i) \odot s_k - 2) + 2$. This entails that $g(\varphi(2i)) = g((2i) \odot s_k - 2) = \tilde{\tilde{g}}((2i) \odot s_k) - 2 = \widetilde{\widetilde{k_{|2}}}((2i) \odot s_k) - 2 = k_{|0}((2i) \odot s_k) - 2 = k_{|1}(2i) \odot s_k - 2 = \tilde{g}(2i) \odot s_k - 2 = g(2i) \odot s_k - 2 = \varphi'(g(2i))$, by Lemmas 5.4.24, 5.4.26, and 5.4.19.

This proves that g is indeed a morphism of TO-plays, hence ends the proof of Theorem 5.4.22.

5.4.3 Restriction to Views

Required: 5.4.2.
Recommended: $\varnothing$.

At last, we show how our functor $F: \mathbb{P}_{A,B} \to \mathbb{E}'(A \vdash B)$ restricts to an equivalence on views, which achieves the construction of our candidate exact square (5.7).

We start with:

Lemma 5.4.27. *If $Y \to U \leftarrow (A \vdash B)$ is a play, then $\prec^*_u$ (Definition 5.2.8) is an order.*

Proof. It suffices to show that $\prec^*$ is antisymmetric, which we do by induction on U. If $U \cong id_{A \vdash B}$, then the result is direct. Otherwise, it is the composite of a move $Y \to M \leftarrow Y'$ and a play $Y' \to U' \leftarrow (A \vdash B)$, and $\prec^*_{U'}$ is an order by induction hypothesis. Now, it suffices to notice that $\prec^*_U$ is an extension of $\prec^*_{U'}$, and that adding the move M can only add pairs $x \prec y$ in which y is an element that is not in U', which cannot break antisymmetry of $\prec^*_U$. □

This allows us to show:

Lemma 5.4.28. *The functor F restricts to a functor $F^{\mathbb{V}}: \mathbb{V}_{A,B} \to (\mathbb{E}^{\mathbb{V}})'(A \vdash B)$.*

Proof. Let $s = (n, f, \varphi)$ be any view. By Lemma 5.4.27, $<^*$ is an order. But for all $i \in n - 1$, using Lemma 5.4.20:

- if i is odd: $(i + 1) \cdot t = i + 1 = i \cdot s$,
- if i is even: $(i+1) \cdot t = \varphi(i+1) + 1$, but because s is a view and $i+1$ is odd, $\varphi(i + 1) = i$, hence $(i + 1) \cdot t = i + 1 = i \cdot s$.

In both cases, we have $i < i + 1$, hence $<^*$ is a total order. □

By left cancellation, $F^{\mathbb{V}}$ is fully faithful (see the beginning of the proof of Lemma 5.3.10). If we prove that it is also essentially surjective, we will obtain:

Theorem 5.4.29. *The restriction $F^{\mathbb{V}}: \mathbb{V}_{A,B} \to (\mathbb{E}^{\mathbb{V}})'(A \vdash B)$ is an equivalence.*

Let us start by showing that any preview $(A \vdash B) \to V$ in $(\mathbb{E}^{\mathbb{V}})'(A \vdash B)$ is isomorphic to some *canonical* preview, in the following sense.

Definition 5.4.30. *A preview $(A \vdash B) \to V$ of length n is* canonical *if and only if*

(i) *it satisfies points (P1)–(P3),*

(ii) *the element of $V(A \vdash B)$ corresponding to $(A \vdash B) \to V$* via *Yoneda is 1, with $1 \odot s_1 = 1$ and $1 \odot t = 2$,*

(iii) *for all $i \in n$, $i \cdot t = i$, $i \cdot s = i + 1$, and $i \cdot \nu = i + 2$,*

where $i \cdot \nu$ denotes the channel created by move i, i.e.,

- *if $i \in V(@_{S,k,m})$, then $i \cdot \nu = i \cdot s \odot t$,*
- *if $i \in V(\Lambda_{S,m})$, then $i \cdot \nu = i \cdot s \odot s_{|S|+1}$, where $|(\Gamma \vdash A)| = |\Gamma|$.*

In words, a preview is canonical when the ith move is represented by i and played by i, and its created player and channel are respectively $i + 1$ and $i + 2$.

Of course, we have:

Lemma 5.4.31. *For all TO-views s, $F(s)$ is canonical.*

Proof. By induction on s. □

Lemma 5.4.32. *Any preview is isomorphic to a unique canonical preview.*

Proof. By induction on the given preview:

- if the preview is isomorphic to $id^{\bullet}_{(A \vdash B)}$, then the result is obvious,
- otherwise, it is the composite of a play $Y \to V \leftarrow X$ and a move $Z \to M \leftarrow X$. By induction hypothesis and Lemma 5.2.10, $Y \to V \leftarrow (A \vdash B)$ is isomorphic to a canonical preview $Y_c \to V_c \leftarrow (A \vdash B)$. To show the result, it suffices to show that M is played by the player y in Y that is mapped to n+1 in V_c. But, since $<_{V_c \bullet M}$ and $<_{V \bullet M}$ have the same structure, if M were played by another player than y, $<^*_{V_c \bullet M}$ would not be a total order, which would contradict the fact that we started from a view. □

Lemma 5.4.33. *In any canonical preview V, for any sequent S and player $i \in V(S)$, we have $|S| = i/2+1$ and furthermore for all $0 \leq k < |S|$, $i \odot s_{k+1} = 2k+1$.*

Proof. By induction on V. □

Proof of Theorem 5.4.29. Lemma 5.4.32 allows us to restrict attention to the claim that canonical previews have antecedents in TO-previews (which, remember, are TO-preplays $s = (n, f, \varphi)$ such that $\lceil s \rceil_n = s$). We again proceed by induction on the given canonical preview V. If V is the identity preview, then an antecedent is given by the empty TO-preview. Now, assume V' is any canonical preview of length n and $V = V' \bullet M$ is a canonical preview. By induction hypothesis, we get $s' = (n, f', \varphi')$ such that $F(s') = V'$.

We will define our candidate antecedent s for V by case analysis on M. In both cases, we will first need to show that s is indeed a TO-preview, and then that $F(s) = V$. By construction, $F(s)$ will be determined by picking a player in V' (i.e., a valid index in $n+1$) and a valid move object in $\mathbb{L}$. By canonicity of V, it will thus be enough to show that these correspond to those of V.

There are two cases, depending on M:

- if $(n+1) \in V(\Lambda_{S,m})$ with $S = (A_1, \ldots, A_N \vdash C)$ and $m \in \sqrt{C}$, then we define $s = (n+1, f, \varphi)$ as the extension of s' by

  - $f(n+1) = m$,
  - $\varphi(n+1) = n$.

  Let us first show that s is a TO-preview. Alternation and P-visibility are trivial, so we only need to verify that s is a justified sequence and that $\lceil s \rceil_{n+1} = s$:

  - $\varphi(n+1) < n+1$: holds trivially;
  - if $\varphi(n+1) = 0$, then $f(n+1) \in \sqrt{B}$: if $\varphi(n+1) = 0$, then by definition $n = 0$, so $S = (A \vdash B)$, hence $C = B$ and $f(n+1) = m \in \sqrt{B}$, as desired;
  - if $\varphi(n+1) \neq 0$, then $f(\varphi(n+1))$ is the parent of $f(n+1)$ in $A \multimap B$: indeed, in that case, $n \in V(@_{(A_1,\ldots,A_N \vdash),k,m'})$ by Lemma 5.4.16, for some $k \in N$ and $m' \in \sqrt{A_k}$, and $C = A_k \cdot m'$, so $m' \vdash_{A \multimap B} m$;
  - finally, $\lceil s \rceil_{n+1} = s$: if $\varphi(n+1) = 0$, then $\lceil s \rceil_{n+1} = (n+1)$ and $n = 0$, and otherwise $\lceil s \rceil_{n+1} = \lceil s' \rceil_{\varphi(n+1)} \cdot (n+1) = \lceil s' \rceil_n \cdot (n+1) = s' \cdot (n+1)$, as desired.

  By construction, the image of s under F is obtained by adding a new move, $n+1$, to $F(s')$ (i.e., to V' by induction hypothesis), with $(n+1) \cdot t = n+1$, of the form $\Lambda_{S_{s',n+1},m}$. This agrees with V, so $F(s) = V$, as desired.

- if $(n+1) \in V(@_{S,k,m})$, with $S = (A_1, \ldots, A_N \vdash)$ and $m \in \sqrt{A_k}$, then we define $s = (n+1, f, \varphi)$ as the extension of s' by

  - $f(n+1) = m$,
  - $\varphi(n+1) = (n+1) \odot s_k - 2$ if $k > 1$,
  - $\varphi(n+1) = (n+1) \odot s_2 - 2$ if $k = 1$.

Let us first show that s is a TO-preview. Alternation is again trivial, and so is P-visibility (because s' is a preview), so we only need to verify that s is a justified sequence such that $\lceil s \rceil_{n+1} = s$:

- $\varphi(n+1) < n+1$: if $k = 1$, then by Lemma 5.4.33 we have $\varphi(n+1) = (n+1) \odot s_2 - 2 = 2+1-2 = 1 < n+1$ because $n+1$ is even; otherwise, by Lemma 5.4.33 again we have $\varphi(n+1) = (n+1) \odot s_k - 2 = 2(k-1)+1-2 = 2k-3$ and $N = (n+1)/2+1$, with $k \leq N$, so $\varphi(n+1) \leq 2((n+1)/2+1)-3 = n < n+1$;
- if $\varphi(n+1) = 0$, then $f(n+1) \in \sqrt{B}$: we in fact have $\varphi(n+1) \neq 0$, because $\varphi(n+1)$ is always odd by Lemma 5.4.33;
- if $\varphi(n+1) \neq 0$, then $f(\varphi(n+1))$ is the parent of $f(n+1)$ in $A \multimap B$: indeed, in that case, we have, by Lemma 5.4.16 and the fact that $\varphi(n+1)$ is odd, that $\varphi(n+1) \in V(\Lambda_{(A_1,\ldots,A_{k-1} \vdash C),m'})$ for some $m' \in \sqrt{C}$, with $A_k = C \cdot m'$; but then $f(\varphi(n+1)) = m'$ is indeed the parent of $f(n+1) = m \in \sqrt{A_k}$;
- finally, $\lceil s \rceil_{n+1} = \lceil s' \rceil_n \cdot (n+1) = s' \cdot (n+1)$, as desired.

By construction, the image of s under F is obtained by adding a new move, $n+1$, to $F(s')$ (i.e., to V' by induction hypothesis), with $(n+1) \cdot t = n+1$, of the form $@_{S_{s',n+1},k',m}$ for some valid k'. The equation agrees with V, so by canonicity it is enough to show $k = k'$.

Again, there are two cases:

- if $f(n+1) \in \sqrt{A}$, then $k' = 1$ but also $k = 1$;
- otherwise $k' = 1 + L_s(n+1)$ by definition of F. Since s is a preview, $I_l = l$ for all l in $n+1$, and so $\varphi(n+1) = I_{2L_s(n+1)-1} = 2L_s(n+1) - 1 = 2k' - 3$. But, as we saw above, $\varphi(n+1) = 2k-3$, so $k = k'$, as desired.

This concludes the proof that $F(s) = V$. □

5.5 The Level of Strategies

Required: 5.4.3, 2.2.6.
Recommended: 2.2.7.

In this chapter's final section we exploit the link we have exhibited between the different categories of plays and views to link the categories of strategies in both approaches. An interesting point is that it is very simple to prove this link using the theory of exact squares.

In the previous section, we have built a commuting square which we reproduce here for convenience:

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r, hook, "\mathsf{i}_{TO}"] \arrow[d, "F^{\mathbb{V}}"'] & \mathbb{P}_{A,B} \arrow[d, "F"] \\
\mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[r, hook, "\mathsf{i}"'] & \mathbb{E}(A \vdash B),
\end{tikzcd}$$

where F is a full embedding and $F^{\mathbb{V}}$ is an equivalence by Theorem 5.3.11.

Lemma 5.5.1. *The square above is exact.*

Proof. By Lemma 2.2.69. □

From which we deduce:

Corollary 5.5.2. *For all arenas A and B, the square*

$$\begin{tikzcd}
\widehat{\mathbb{V}_{A,B}} \arrow[r, hook, "\Pi_{\mathsf{i}_{TO}}"] & \widehat{\mathbb{P}_{A,B}} \\
\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \arrow[u, "\Delta_{F^{\mathbb{V}}}"] \arrow[r, hook, "\Pi_{\mathsf{i}}"'] & \widehat{\mathbb{E}(A \vdash B)} \arrow[u, "\Delta_F"']
\end{tikzcd}$$

commutes up to isomorphism.

Restriction along $F^{\mathbb{V}}$ induces an equivalence between behaviours over $(A \vdash B)$ and TO-behaviours over (A, B).

Restriction along F induces a functor from strategies over $(A \vdash B)$ to TO-strategies over (A, B), which restricts to an equivalence on innocent strategies.

The moral of this result is that our views and plays faithfully represent Tsukada and Ong's. Indeed, both notions of behaviour essentially coincide and moreover, although our categories of plays are slightly richer, our innocent strategies restrict to theirs and furthermore their process of extending behaviours to innocent strategies coincides with ours up to this restriction.

This also implies that these equivalences are compatible with *innocentisation*, which takes a possibly non-innocent strategy and canonically turns it into an innocent strategy. This is done by *sheafification*, which, in our case, is the functor $\prod_{\mathsf{i}} \circ \Delta_{\mathsf{i}}$. Indeed, consider the diagram:

$$\begin{tikzcd}
\mathbb{E}(A \vdash B) & \mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[l, "\Delta"'] \arrow[r, "\Pi"] & \mathbb{E}(A \vdash B) \\
\mathbb{P}_{A,B} \arrow[u, "\Delta"] & \mathbb{V}_{A,B} \arrow[l, "\Delta"] \arrow[u, "\Delta"'] \arrow[r, "\Pi"'] & \mathbb{P}_{A,B}. \arrow[u, "\Delta"']
\end{tikzcd}$$

Remember that, when drawing diagrams using $\prod$'s, $\sum$'s, and Δ's, we only draw the underlying diagram (which in particular means that Δ arrows point in the "wrong" direction). If we take the top path in the diagram above, we first innocentise the strategy (inside the string diagrammatic setting) and then take the equivalent TO-strategy. If we take the bottom path, we first take the equivalent TO-strategy, and then innocentise it (inside Tsukada and Ong's setting). But the left-hand square commutes because the underlying diagram (without the Δ's) does, and the right-hand side one commutes by exactness.

5.6 Perspectives

One may notice that a crucial point is left untreated here: Tsukada and Ong can compose strategies, while it is not clear how to do this directly in our setting. Two steps are required to compose strategies, parallel composition and hiding: the first executes two strategies in parallel, and the second one hides the middle arena. While parallel composition is easy to manipulate in our setting because our game is intrinsically multi-party, hiding could admittedly be more difficult to handle.

Another point, which we already mentioned at the end of Chapter 4 is that we do not use quite the same definition of views in this setting as in our other models. As we explained when we defined views, we cannot define views as in our other models and use our proof of fibredness. If we used the same technique as in our previous models, we would break persistence, which we proved to be necessary for fibredness (Theorem 4.4.30) to hold (see the remark after the proof of Lemma 4.4.32). We thus cannot hope to have the same definition of view for our model of HON games as in our model of the π-calculus using this notion of signature. This point makes us think that maybe the notion of signature we used is a bit too restrictive. It is not yet clear to us how to solve this problem.

Another problem with this setting is that interaction sequences are not faithfully modelled by plays on the pushout

$$\begin{tikzcd}
B \arrow[r, "t"] \arrow[d, "s_1"'] \arrow[dr, phantom, "\ulcorner", very near end] & (A \vdash B) \arrow[d, "\mathrm{inl}"] \\
(B \vdash C) \arrow[r, "\mathrm{inr}"'] & (A \vdash B \vdash C).
\end{tikzcd}$$

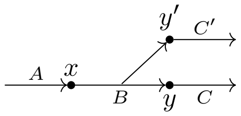
Indeed, using the same techniques as in this chapter, it is not hard to show that there is a full embedding $\mathbb{I}_{A,B,C} \to \mathbb{E}(A \vdash B \vdash C)$, where $\mathbb{I}_{A,B,C}$ is the category of interaction sequences in Tsukada and Ong's model. However, there are objects in $\mathbb{E}(A \vdash B \vdash C)$ that do not make much sense from a computational point of view, as we try to explain now. Let us call x_0 the initial $(A \vdash B)$ player and y_0 the initial $(B \vdash C)$ player, and let us give ourselves terms $f\colon A \to B$ and $g\colon B \to C$ of some language (say a non-deterministic λ-calculus) interpreted as strategies S_f on $(A \vdash B)$ and S_g on $(B \vdash C)$. After some moves have been played on the position $(A \vdash B \vdash C)$, one can divide the final position into two "teams" of players: those players x such that $x_0 \prec^* x$, and those such that $y_0 \prec^* x$. The first team is made of those players that represent some ongoing or frozen computation of f, while the second team is made of players that represent computations of g. Some moves make complete sense from a computational point of view:

- β moves between two players: it can be proved that such a move is always played between two players that are on different teams, and they correspond to an interaction between f and g, just like playing on the middle arena in an interaction sequences in traditional game semantics,

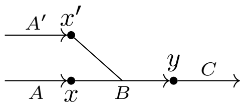
- Λ and @ moves with the outside (i.e., on sub-arenas of A or C): they correspond to interacting with the outside, just like playing on the two extremal arenas in traditional game semantics,

- Λ moves on B: it can be proved by induction that such moves may only be played by players of the first team, and they correspond once again to interaction with the outside, since program fragments other than g may call f.

The last point is slightly more controversial, but it makes sense if we see $(A \vdash B \vdash C)$ as a sub-position of



and the Λ move as a sub-move of a β move between x and y'. However, @ moves on B do not make sense from a computational point of view, because they represent g asking f for a value, but somehow ending up interacting with the outside while doing so. This corresponds to the fact that positions of the form



do not really make sense in our game, and that y must thus necessarily synchronise with x if they want to play @ on B.

An idea that could possibly solve this problem would be to define plays not as presheaves, but as sheaves for a Grothendieck topology stating that β moves are covered by @ moves and the presence of a player that could play Λ to synchronise with that @ move.

Chapter 6

Composing Non-Deterministic Strategies

6.1 Motivation

Game models offer an intuitive and simple semantical framework that nonetheless provides fully abstract models for a variety of languages. However, the proofs involved are usually very complex with respect to the definitions. Let us give the example of two fundamental properties of these models that are not straightforward to prove. Their proofs also depend on the particular variant of games studied, which raises the issue of finding a theory that would be general enough to encompass a large number of these models, but also powerful enough to factor out these proofs.

The first one is maybe the most fundamental property of a game model (or any denotational model). It is that games and strategies (or the interpretations of programs in the case of denotational models) form a category and that composition of strategies coincides with composition of programs. Even the fact that games and strategies form a category is not straightforward to prove.

Another fundamental notion is that of *innocence*, which is the fact that the behaviour of a program should only depend on what it "sees" of the environment (typically, how the environment interacts with the program, the values it sends to the program, etc.), and not on the rest of the execution (how the environment interacts with other program fragments, *how* it computes the values it sends to the program, etc.). This is of course not a reasonable hypothesis in general, because the program may for example keep a pointer or reference to some values that change depending on how a value is computed, but a key result in game semantics is that innocent strategies exactly correspond to the interpretations of purely functional programs. It is therefore crucial to verify that innocent strategies form a subcategory of the category of games and strategies.

The work we discuss in this chapter started as an attempt at taking the proof of Tsukada and Ong [97] that their arenas and innocent strategies form a category and recasting it using different tools that we thought to be more fit for this purpose. By using high-level tools, we get the same result with simpler proofs, and are also able to precisely extract the hypotheses used in the proofs without drowning in details about how exactly plays are represented.

This led us to a surprisingly broad framework, that may be applied to streamline the proofs that composition of strategies is associative and unital in different settings. In non-technical terms, we first define an abstract gadget, which we call a *game setting* (which basically represents plays) and show how that gadget gives rise to a notion of strategy, as well as copycat strategies and a composition functor. We then show that, under mild hypotheses, games and non-deterministic strategies form a category. Under some more hypotheses, we prove that non-deterministic, innocent strategies form a subcategory of strategies. A game semanticist should be doubtful at this point, as it is well known that non-deterministic, innocent strategies do not compose, but the notion of strategy we use here is slightly different from the traditional one, though there is an intuitive correspondence between the two. We then treat composition of "traditional" strategies and show that non-deterministic strategies form a category and that restraining to *deterministic*, innocent strategies gives a subcategory. In passing, we can pinpoint where our method fails if we try to show that non-deterministic, innocent strategies compose.

All the while, we make sure to relate composition of our strategies, which are not exactly traditional, with composition of traditional ones, and show in which sense the two correspond.

6.1.1 The Main Ideas

We define strategies as presheaves over a category of plays. Such a strategy S accepts the play p if $S(p)$ is non-empty and rejects it otherwise. We may think of $S(p)$ as the set of states S may be in once it has played p (if S cannot be in any state after playing p, then S refuses to play p altogether). The fact that S may be in several states after playing p reflects the fact that S is inherently non-deterministic. If there is a morphism $p \to p'$ (which typically means that p is a prefix of p'), then the presheaf structure induces a function from $S(p')$ to $S(p)$, so that, if p' is accepted, then so is p. This notion of strategy is not exactly traditional because a strategy may accept a play in several ways (the cardinal of $S(p)$), while a traditional strategy barely accepts or rejects plays. We thus also define "traditional" strategies as presheaves to 2, the ordinal $0 \to 1$ seen as a category. There is an obvious way to translate presheaves as presheaves to 2, given by collapsing all non-empty sets to 1 and the empty set to 0. We thus have two fundamentally different types of strategies: general strategies (or g-strategies) and boolean ones (or b-strategies).

Our first contribution is to define composition of g-strategies and copycat g-strategies as polynomial functors [62, 35] (i.e., composites of $\prod$'s, $\sum$'s, and Δ's) and to show that these definitions agree with Tsukada and Ong's. Our definition of composition is conceptually simple, in the sense that it can directly be read as "the composite $\sigma_1;\sigma_2$ of two strategies accepts a play p on the pair of games (A, C) if there is an interaction sequence u on (A, B, C) that projects to p and whose projections to (A, B) and (A, C) are accepted by σ_1 and σ_2 respectively". The use of polynomial functors also gives us the opportunity to use Guitart's theory of *exact squares* (see Section 2.2.6), which shifts the focus from the categories of presheaves to the base categories, leading to simpler proofs. We thus show that, under some hypotheses on the game setting, composition of g-strategies is associative and unital. We also show how this implies that composition of b-strategies is associative and unital by exhibiting links between compositions in both settings.

We then set out to instantiate our framework on different game settings, such as traditional HON games and AJM games, different variations on HON games, and Tsukada and Ong's games. We also show how the hypotheses imposed on game settings are actually necessary by showing that Blass games, a classic example of strategies whose composition is not associative, do not verify them.

Finally, we treat the case of innocent strategies. Traditionally, a strategy S is innocent when it accepts a play p if and only if it accepts all the *views* (a particular type of play) that embed into it. Hirschowitz has argued [50] that this idea may be recast as a *sheaf condition*, by viewing a play as covered by its views. An innocent g-strategy is simply a sheaf for the topology induced by the embedding of views into plays. By adding structure describing the embedding of views into plays to our game settings, we can express innocence and study composition of innocent strategies. We show that composition of innocent g-strategies is associative under some conditions. To study composition of traditional, deterministic, innocent b-strategies, we need to take into account the fact that the notion of morphism of plays may have changed: typically, we want to infer that games and innocent b-strategies form a category for HON games from the fact that Tsukada and Ong's g-strategies form a category, but Tsukada and Ong consider a richer notion of morphism than prefix inclusion, which is the notion of morphisms of plays in HON games. We thus have more notions of innocence: p-innocence if morphisms form a poset (e.g., prefix inclusion) and c-innocence if morphisms form a proper category. From the fact that innocent cg-strategies form a category, we derive that innocent pg-strategies form a category. We cannot infer that innocent pb-strategies form a category (because they do not), but manage to show that it is the case for deterministic, innocent pb-strategies.

6.1.2 A Technical Point

Required: 2.2.6.
Recommended: ∅.

As everything is expressed in terms of polynomial functors, the theory of exact squares provides a powerful toolbox to show that diagrams commute (up to isomorphism). However, this theory is limited to squares of one of the forms

$$\begin{tikzcd}
\widehat{A} & \widehat{B} \arrow[l, "\Delta_T"'] \arrow[d, "\Sigma_V"] \\
\widehat{C} \arrow[u, phantom] & \widehat{D} \arrow[l, "\Delta_U"]
\end{tikzcd}
\quad \text{or} \quad
\begin{tikzcd}
\widehat{A} \arrow[r, "\Pi_T"] & \widehat{B} \\
\widehat{C} \arrow[u, "\Delta_S"] \arrow[r, "\Pi_U"'] & \widehat{D}. \arrow[u, "\Delta_V"']
\end{tikzcd}$$

with $\Sigma_S \colon \widehat{A} \to \widehat{C}$ on the left of the first square.

It is even easier to show that squares that only involve one of $\prod$, $\sum$, or Δ commute, since it suffices to show that the underlying diagram (the one on the base categories, rather than the presheaf categories) commutes. But we will also need a lemma to make left and right Kan extension commute, which we now discuss.

Consider an exact square as below left. Because it is exact, the middle natural transformation is an isomorphism, so it can be inverted, which gives rise by the mate calculus to a square as below right.

$$\begin{tikzcd}
A \arrow[r, "T"] \arrow[d, "S"'] \arrow[dr, phantom, "\overset{\varphi}{\Longrightarrow}"] & C \arrow[d, "V"] & \widehat{A} \arrow[d, "\Sigma_S"'] & \widehat{C} \arrow[l, "\Delta_T"'] \arrow[d, "\Sigma_V"] \arrow[dl, phantom, "\Downarrow \Sigma_\varphi"] & \widehat{A} \arrow[r, "\Pi_T"] \arrow[d, "\Sigma_S"'] \arrow[dr, phantom, "\overset{\tilde{\varphi}}{\Longleftarrow}"] & \widehat{C} \arrow[d, "\Sigma_V"] \\
B \arrow[r, "U"'] & D & \widehat{B} & \widehat{D} \arrow[l, "\Delta_U"] & \widehat{B} \arrow[r, "\Pi_U"'] & \widehat{D}
\end{tikzcd}$$

Definition 6.1.1. *An exact square is* distributive *when $\tilde{\varphi}$ is an isomorphism.*

Let us consider any functors $A \xrightarrow{S} B \xrightarrow{U} D$ with S a discrete fibration and U fully faithful. Remember that, in Section 2.2.5, we denoted by $\partial^*(S)$ the presheaf that corresponds to S through the equivalence of presheaves and discrete fibrations. Let $C = \mathrm{el}(\prod_U(\partial^*(S))$ denote the category of elements of the right Kan extension of $\partial^*(S)$ along U^{op}, and let V denote its projection to D. Because U is fully faithful, the counit ε of the adjunction $\Delta_U \dashv \prod_U$ is an isomorphism, so the left-hand square of

$$\begin{tikzcd}
A \arrow[r, "\varepsilon_S"] \arrow[d, "S"'] & \cdot \arrow[r] \arrow[d, "\Delta_U(V)"] & C \arrow[d, "V=\Pi_U(S)"] \\
B \arrow[r, equal] & B \arrow[r, "U"'] & D
\end{tikzcd}$$

is a pullback. The right-hand square is a pullback by Lemma 2.2.58, so the rectangle also is by the pullback lemma. By Lemma 2.2.70, it is exact, so we may wonder whether or not it is distributive. This square is called the *local pushforward square* of S and U.

Lemma 6.1.2. *Local pushforward squares are distributive.*

Proof. We first prove the corresponding result for the equivalent categories of discrete fibrations. There, because S and V are discrete fibrations, $\sum_S$ becomes post-composition with S, and similarly for Σ_V. So the result essentially states that for any discrete fibration $P: X \to A$, $\prod_U(\sum_S(P)) \cong \prod_U(SP) \cong \prod_U(S)\prod_T(P) \cong V \circ \prod_T(P) \cong \sum_V(\prod_T(P))$, where the only non-obvious point is that $\prod_U(SP) \cong \prod_U(S)\prod_T(P)$. To prove that this isomorphism holds, we show that $\prod_U(S)\prod_T(P)$ enjoys the same universal property as $\prod_U(SP)$. In other words, we show that, for all discrete fibrations $Q: Y \to D$, $[\Delta_U(Q), SP] \cong [Q, \prod_U(S)\prod_T(P)]$, naturally in Q.

We first show how to map each morphism F in $[Q, \prod_U(S)\prod_T(P)]$ to a morphism in $[\Delta_U(Q), SP]$. Given such a morphism, we have a diagram as the solid part of

$$\begin{tikzcd}
 & X \arrow[rr] \arrow[dd, "P"'] \arrow[dr, phantom, very near start, "\lrcorner"] & & Y \arrow[dd, "\Pi_T(P)"] \\
Z' \arrow[ur, dashed] \arrow[rr] \arrow[ddr, "\Delta_U(Q)"'] \arrow[dr, phantom, very near start, "\lrcorner"] & & Z \arrow[ur, "F"'] \arrow[ddr, "Q"] & \\
 & A \arrow[rr, "T"] \arrow[d, "S"'] \arrow[dr, phantom, very near start, "\lrcorner"] & & C \arrow[d, "\Pi_U(S)"] \\
 & B \arrow[rr, "U"'] & & D
\end{tikzcd}$$

and want to define the dashed arrow. (We can build the top square because fully faithful functors are the right class of a factorisation system, so are stable under pullbacks, and U is fully faithful.) We could simply define it by the universal property of pullback (and that would indeed be the right construction), but we prefer using adjunctions $\Delta \dashv \prod$, because it will make the proof more symmetric. We have that $\prod_T(P)F$ is a morphism from Q to $\prod_U(S)$, so it corresponds by adjunction to a morphism P' from $\Delta_U(Q)$ to S, and

$$\begin{tikzcd}
Z' \arrow[r] \arrow[d, "P'"'] & Z \arrow[d, "\Pi_T(P)F"] \\
A \arrow[r, "T"'] & C
\end{tikzcd}$$

is a pullback by the pullback lemma, so that P' is isomorphic to $\Delta_T(\prod_T(P)F)$. But now, F is a morphism from $\prod_T(P)F$ to $\prod_T(P)$, which yields by adjunction a morphism $\tilde{F}$ from $\Delta_T(\prod_T(P)F)$ to P, hence from P' to P. We thus get that $\tilde{F}$ is a morphism from $\Delta_U(Q)$ to SP, and this mapping is monic and natural because all arrows arise from the use of natural isomorphisms. The proof that any morphism in $[\Delta_U(Q), SP]$ may be mapped naturally to one in $[Q, \prod_U(S)\prod_T(P)]$ is similar.

What we have shown here is that

$$\begin{tikzcd}
\mathsf{DFib}_A \arrow[r, "\Pi_T"] \arrow[d, "\Sigma_S"'] & \mathsf{DFib}_C \arrow[d, "\Sigma_V"] \\
\mathsf{DFib}_B \arrow[r, "\Pi_U"'] & \mathsf{DFib}_D
\end{tikzcd}$$

commutes up to isomorphism. We now transfer this result, replacing DFib_X with $\widehat{X}$ through the following diagram:

$$\begin{tikzcd}
\widehat{A} \arrow[rr, "\Pi_T"] \arrow[dr, "\Sigma_S"'] \arrow[dd] & & \widehat{C} \arrow[dr, "\Sigma_V"] \arrow[dd] & & \\
& \widehat{B} \arrow[rr, "\Pi_U"] \arrow[dd] & & \widehat{D} \arrow[dd] \arrow[dr, equal] & \\
\mathsf{DFib}_A \arrow[rr, "\Pi_T"] \arrow[dr, "\Sigma_S"'] & & \mathsf{DFib}_C \arrow[dr, "\Sigma_V"] & & \widehat{D}. \\
& \mathsf{DFib}_B \arrow[rr, "\Pi_U"'] & & \mathsf{DFib}_D \arrow[ur] &
\end{tikzcd}$$

Our goal is to prove that the top part of the diagram commutes up to isomorphism, which we prove by showing that everything else does. We just proved that bottom square commutes up to isomorphism, and the final triangle also does because (π, ∂^*) forms an equivalence, so we are left with showing that the vertical squares commute up to isomorphism.

For any functor $F\colon \mathbb{C} \to \mathbb{D}$, we know that the left-hand square below commutes up to isomorphism, so by pre- and post-composing with ∂^*, we get that the right-hand one does too, using the fact that (π, ∂^*) form an equivalence.

$$\begin{tikzcd}
\widehat{\mathbb{C}} \arrow[d, "\pi"'] & \widehat{\mathbb{D}} \arrow[l, "\Delta_F"'] \arrow[d, "\pi"] \\
\mathsf{DFib}_{\mathbb{C}} & \mathsf{DFib}_{\mathbb{D}} \arrow[l, "\Delta_F"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\mathbb{C}} & \widehat{\mathbb{D}} \arrow[l, "\Delta_F"'] \\
\mathsf{DFib}_{\mathbb{C}} \arrow[u, "\partial^*"] & \mathsf{DFib}_{\mathbb{D}} \arrow[l, "\Delta_F"] \arrow[u, "\partial^*"']
\end{tikzcd}$$

Now, using the fact that $\sum_F \dashv \Delta_F$ and $\pi \dashv \partial^*$ form adjunctions and uniqueness of left adjoint up to isomorphism, we get that the left-hand square below commutes up to isomorphism, and similarly for the right-hand one, using $\Delta_F \dashv \prod_F$, $\partial^* \dashv \pi$, and uniqueness of right adjoint.

$$\begin{tikzcd}
\widehat{\mathbb{C}} \arrow[rr, "\Sigma_F"] \arrow[d, "\pi"'] & & \widehat{\mathbb{D}} \arrow[d, "\pi"] \\
\mathsf{DFib}_{\mathbb{C}} \arrow[rr, "\Sigma_F"'] & & \mathsf{DFib}_{\mathbb{D}}
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\mathbb{C}} \arrow[rr, "\Pi_F"] \arrow[d, "\pi"'] & & \widehat{\mathbb{D}} \arrow[d, "\pi"] \\
\mathsf{DFib}_{\mathbb{C}} \arrow[rr, "\Pi_F"'] & & \mathsf{DFib}_{\mathbb{D}}
\end{tikzcd}$$

Applying this result to all four vertical faces concludes the proof. □

6.2 Polynomial Functors for Abstract Game Semantics

The main idea behind game settings is that they describe plays with two or more players, and the link that must exist between these notions of plays. We organise this information into a categorical structure from which we abstractly get two notions of strategies: the presheaf-based one and essentially the standard one. We then prove that games and strategies form a category, for both notions of strategies, and exhibit a link between composition of both types of strategies.

The construction of a typical game model relies on the definition of notions of plays between increasingly many players. There is first a notion of *game*. Each game A involves two players O (Opponent) and P (Proponent), and features in particular a set of *plays* $\mathbb{P}_A$, which may be endowed with the prefix ordering or with a more sophisticated notion of morphism, thus forming a category of plays.

Example 6.2.1. *In games based on arenas (see Section 2.1 for some examples), a game is an arena A taken from the set $\mathbb{A}$ of all arenas. Plays on such arenas are typically defined as alternating justified sequences of elements of A of even length (this is enough to understand the intuitions). This definition is however not broad enough (see the definition of $\mathbb{P}_{A,B}$ in the case of HON-style games just below), so, if we consider HON-style games, $\mathbb{P}_A$ will be given by potentially non-alternating justified sequences of elements of A of any finite length, as in Definition 2.1.4.*

The crucial step to view strategies as morphisms is to consider the *arrow* game $A \multimap B$, which intuitively describes the interaction of a middle player M acting as Opponent against a left player L and as Proponent against a right player R, as in

$$\begin{tikzcd}
L & & M & & R \\
& \mathbb{B} \arrow[rr] & & \mathbb{B} & \\
& & & q^R & \\
& q^M & & & \\
& \mathsf{t}^L & & & \\
& & & \mathsf{f}^M. &
\end{tikzcd} \tag{6.1}$$

In this example, M plays like the negation function on booleans: R asks its return value by playing the move q^R; M in turn asks L for an argument by playing q^M, to which L answers true by playing t^L; M eventually answers the original question by playing f^M (in Section 2.1.1, L and R are seen as a single player, since they are both played by the environment).

However, there is a subtlety: one often needs to restrict attention to a certain subcategory $\mathbb{P}_{A,B} \hookrightarrow \mathbb{P}_{A \multimap B}$. There must be projections to $\mathbb{P}_A$ and $\mathbb{P}_B$, first because it makes the whole categorical formalisation simpler, but also because we do not get the right notion of copycat strategy otherwise (for a technical reason). Note that this is not simply a trifling detail, as it is the very reason why we take all justified sequences in $\mathbb{P}_A$ (instead of the more traditional choice of alternating justified sequences of even length).

Example 6.2.2. *In HON-style games, $\mathbb{P}_{A,B}$ would consist of alternating sequences of even length of $\mathbb{P}_{A \multimap B}$. The projections of a play in $\mathbb{P}_{A,B}$ to A and B may not be alternating or have even length, so it is crucial to be liberal in the choice of $\mathbb{P}_A$.*

For composition of strategies, the situation (6.1) is then scaled up to combinations of two such situations in which a first middle player M_1 plays on the right with a second one, say M_2, as in

$$\begin{tikzcd} L & M_1 & M_2 & R \\ \mathbb{B} \arrow[r] & \mathbb{B} \arrow[r] & \mathbb{B}. \end{tikzcd} \tag{6.2}$$

Plays in such combinations are standardly called *interaction sequences*, and typically form a subcategory $\mathbb{P}_{A,B,C} \hookrightarrow \mathbb{P}_{(A \multimap B) \multimap C}$. An important point is that interaction sequences admit projections to $\mathbb{P}_{A,B}$, $\mathbb{P}_{B,C}$, and $\mathbb{P}_{A,C}$, which satisfy the obvious equations with respect to further projections to $\mathbb{P}_A$, $\mathbb{P}_B$, and $\mathbb{P}_C$, e.g., the following square commutes:

$$\begin{tikzcd} \mathbb{P}_{A,B,C} \arrow[r] \arrow[d] & \mathbb{P}_{A,C} \arrow[d] \\ \mathbb{P}_{B,C} \arrow[r] & \mathbb{P}_C. \end{tikzcd}$$

Example 6.2.3. *In HON games, $\mathbb{P}_{A,B,C}$ typically consists of alternating justified sequences on $(A \multimap B) \multimap C$ which end in A or C and whose projections to $A \multimap B$ and $B \multimap C$ are plays.*

Finally, in order to prove associativity of composition, one defines *generalised interaction sequences* as a subcategory $\mathbb{P}_{A,B,C,D} \hookrightarrow \mathbb{P}_{((A \multimap B) \multimap C) \multimap D}$, again with projections satisfying the obvious equations.

6.2.1 Plays as a Category-Valued Presheaf

Required: 2.2.5.
Recommended: ∅.

Let us now organise all this data ($\mathbb{P}_A$, $\mathbb{P}_{A,B}$, $\mathbb{P}_{A,B,C}$, $\mathbb{P}_{A,B,C,D}$, and the existence of projections) into a simple categorical structure. First, as suggested by our notation, for all lists $L = (A_0, \dots, A_{n-1})$ of games, we construct a category $\mathbb{P}_L$.

Example 6.2.4. *In the HON case, we may take $\mathbb{P}_L$ to consist of alternating justified sequences on $(\ldots(A_0 \multimap A_1) \multimap \ldots) \multimap A_{n-1}$ whose projection to each $A_i \multimap A_{i+1}$ is a play, and which end in A_0 or A_{n-1}, the leftmost and rightmost arenas.*

For the same reasons as before, we need projections from $\mathbb{P}_L$ to $\mathbb{P}_{L'}$, where L' contains one less arena than L. If L is the list $A_0, \ldots, A_{n-1}$, we define L^{-k} to be the list $A_0, \ldots, A_{k-1}, A_{k+1}, \ldots, A_{n-1}$. We thus need projections $\delta_k \colon \mathbb{P}_L \to \mathbb{P}_{L^{-k}}$ (for "delete k"), for all $k \in [n]$.

A similar construction, relevant for defining copycat strategies is *insertions* $\iota_k \colon \mathbb{P}_L \to \mathbb{P}_{L^{+k}}$ (for "insert k"), where $k \in [n]$ and L^{+k} denotes $A_0, \ldots, A_{k-1}, A_k, A_k, A_{k+1}, \ldots A_{n-1}$. For example, there is an insertion ι_1 from $\mathbb{P}_{A,B}$ to $\mathbb{P}_{A,B,B}$. Intuitively, this functor maps any play u in $\mathbb{P}_{A,B}$ to the interaction sequence in $\mathbb{P}_{A,B,B}$ which duplicates all moves on B. So in a situation like (6.2), M_2 would act as a "proxy" between M_1 and R, repeating M_1's moves to R and conversely. For a precise definition and an example in the case of HON games, see Section 6.3.1.

We can express the existence of all insertion and deletion morphisms in a simple categorical structure. Let Δ be the simplex category, i.e., the category whose objects are finite ordinals $[n]$ and whose morphisms $[n] \to [m]$ are order-preserving functions from n to m. Let us call i the embedding of Δ into Set. Recall that $\mathbb{A}$ is the set of all games. Then $\Delta/\mathbb{A}$ (or, more precisely, $\mathsf{i} \downarrow \ulcorner \mathbb{A} \urcorner$) has finite lists of games as objects, i.e., maps $L \colon [n] \to \mathbb{A}$ for some n, and as morphisms $(n, L) \to (n', L')$ all monotone maps f making the following triangle commute:

$$\begin{tikzcd}
{[n]} \arrow[rr, "f"] \arrow[dr, "L"'] & & {[n']} \arrow[dl, "L'"] \\
& \mathbb{A}. &
\end{tikzcd}$$

Example 6.2.5. *We will use the following morphisms to define δ_k and ι_k. Let $\mathsf{d}^n_k \colon [n] \to [n+1]$ miss $k \in [n]$, i.e., $\mathsf{d}^n_k(i) = i$ for $i < k$ and $\mathsf{d}^n_k(j) = j + 1$ for $j \geq k$. E.g., d^2_1 yields a map $(A, C) \to (A, B, C)$ for all games A, B, C. Similarly, consider $\mathsf{i}^n_k \colon [n+1] \to [n]$ which collapses $k \in [n] \subseteq [n+1]$ and $k + 1 \in [n+1]$. E.g., for $n = 2$ and $k = 0$, it yields a map $(A, A, B) \to (A, B)$ for all A and B.*

As promised, this yields a way to organise the categories of plays involved in a typical game model into a coherent categorical structure: we will show below that, for quite a few game models, the assignment $L \mapsto \mathbb{P}_L$ induces a category-valued presheaf on $\Delta/\mathbb{A}$, i.e., a functor $(\Delta/\mathbb{A})^{op} \to \mathsf{Cat}$. Furthermore, the maps δ_k and ι_k introduced earlier will respectively be given by $\mathbb{P}(\mathsf{d}_k)$ and $\mathbb{P}(\mathsf{i}_k)$.

In the following, we will only need to use this structure up to lists of length 4:

Definition 6.2.6. *For any p in $\mathbb{N}$ and set $\mathbb{A}$, let $\mathbb{A}_{[1,p]}$ denote the full subcategory of $\Delta/\mathbb{A}$ spanning lists L of length up to p.*

In the next sections, we will define strategies, composition and copycat strategies abstractly, based on the category-valued presheaf $\mathbb{P}$ on $\mathbb{A}_{[1,4]}$. This is quite demanding, but we are rewarded with a high-level view of composition, which yields abstract proofs of associativity and unitality, under mild hypotheses on $\mathbb{P}$. We will define a game setting to consist of a set $\mathbb{A}$ and a category-valued presheaf satisfying these hypotheses. Of course, it will remain to show, for each considered game model, that

- plays indeed form a game setting and
- the standard definitions of composition and copycat strategies agree with the abstract ones.

6.2.2 Copycats and Composition as Polynomial Functors

Required: 6.2.1, 2.2.4.
Recommended: $\varnothing$.

Let us now start our reconstruction of a game model from an arbitrary category-valued presheaf $\mathbb{P}$ on $\mathbb{A}_{[1,4]}$. Our first step is to define strategies. Standardly, a strategy $\sigma\colon A \to B$ is a prefix-closed set of plays in $\mathbb{P}_{A,B}$ (generally required to be non-empty). Equivalently, it is a functor $\mathbb{P}_{A,B}^{op} \to 2$, where, we recall, 2 denotes the ordinal $0 \to 1$ viewed as a category. In Tsukada and Ong's model [97], $\mathbb{P}_{A,B}$ is a proper category, and strategies are generalised to *presheaves* on $\mathbb{P}_{A,B}$, i.e., functors $\mathbb{P}_{A,B}^{op} \to \mathsf{Set}$. This is indeed a generalisation because 2 embeds into Set (more on this in Section 6.2.4).

The basis of our approach will be the general notion:

Definition 6.2.7. *Let the category of* strategies *from A to B be $\widehat{\mathbb{P}_{A,B}}$. The category of* boolean strategies *is $\widetilde{\mathbb{P}_{A,B}}$, the category of functors $\mathbb{P}_{A,B}^{op} \to 2$.*

The next step in our reconstruction of a game model from $\mathbb{A}$ and $\mathbb{P}$ is to define identities and composition.

Definition 6.2.8. *A functor is* polynomial *if it is isomorphic to some finite composite of functors of the form Δ_F, $\prod_F$ and $\sum_F$.*

This definition may be seen as a generalisation of Fiore's [35], and would turn out to be the same if his polynomial functors are ever proved to be closed under composition.

Remember that we will use two notations for polynomial functors:

- the "full" notation, which we will mainly use in the definitions: it is simply to write the functor as, say, $\widehat{\mathbb{A}} \xrightarrow{\Pi_F} \widehat{\mathbb{B}} \xrightarrow{\Sigma_G} \widehat{\mathbb{C}} \xrightarrow{\Delta_H} \widehat{\mathbb{D}}$,
- a lighter notation, which we will mainly use in the proofs, in which the underlying functors and presheaf categories are left implicit, so the same functor as above would be written $\mathbb{A} \xrightarrow{\Pi} \mathbb{B} \xrightarrow{\Sigma} \mathbb{C} \xleftarrow{\Delta} \mathbb{D}$.

In particular, note that, in the second notation, we draw arrows in the direction of the underlying arrow, so the Δ arrows are reversed, like $\mathbb{C} \xleftarrow{\Delta} \mathbb{D}$. This may be more difficult to read in general, but our diagrams will always be written from left to right (and most from top-left to bottom-right), so they should not be too difficult to read. The real point of this notation is that it shifts the focus from the categories of presheaves to the underlying ones, which helps a lot, since our proofs will all focus on the level of the base categories, thanks to exact squares and local pushforward squares.

As a warm-up before considering composition, we would like to start with our abstract definition of copycat strategies. A natural way to define the copycat strategy $id_A\colon A \to A$ is to decree that it accepts all plays in $\mathbb{P}_{A,A}$ that are in the image of the insertion functor $\iota_0\colon\mathbb{P}_A \to \mathbb{P}_{A,A}$. Indeed, recalling (6.1) and according to the discussion about insertions right after Example 6.2.4, such plays are precisely those in which M acts as a proxy between L and R, which agrees with the standard definition of copycat strategies.

This definition has the advantage of concreteness, but let us give an equivalent, polynomial definition. Because an object of a category $\mathbb{C}$ is the same as a functor $1 \to \mathbb{C}$, we may define id_A as a functor $1 \to \widehat{\mathbb{P}_{A,A}}$. Furthermore, 1 is a presheaf category: indeed it is $\widehat{\varnothing}$, presheaves over the empty category. So we may view the copycat strategy over A as a functor $\widehat{\varnothing} \to \widehat{\mathbb{P}_{A,A}}$. In order to present it as a polynomial functor, we will need to assume that the insertion functor $\iota_0\colon\mathbb{P}_A \to \mathbb{P}_{A,A}$ is a discrete fibration, whose relevant property here is the characterisation of left extension along them, as given by Lemma 2.2.27. Here is our polynomial presentation of copycat:

Proposition 6.2.9. *If the insertion functor $\iota_0\colon\mathbb{P}_A \to \mathbb{P}_{A,A}$ is a discrete fibration, then the functor $\widehat{\varnothing} \xrightarrow{\Pi_!} \widehat{\mathbb{P}_A} \xrightarrow{\Sigma_{\iota_0}} \widehat{\mathbb{P}_{A,A}}$ is isomorphic the copycat strategy id_A.*

Proof. First, because $\prod_!$ is right adjoint to $\Delta_!$, it preserves the terminal object (which is the unique object of $\widehat{\varnothing}$), hence maps 1 to the terminal presheaf on $\mathbb{P}_A$, defined to map any play in $\mathbb{P}_A$ to 1. So we reduce to showing that $id_A \cong \sum_{\iota_0}(1)$.

By Lemma 2.2.27, we know that $(\sum_{\iota_0} X)(p) \cong \sum_{\iota_0(q)=p} X(q)$ for any presheaf X on $\mathbb{P}_A$ and $p \in \mathbb{P}_{A,A}$. So in particular when $X = 1$ we get $\sum_{\iota_0}(1)(p) \cong \sum_{\iota_0(q)=p} 1 \cong \begin{cases} 1 & \text{if } p \in \mathrm{Im}(\iota_0) \\ \varnothing & \text{otherwise,} \end{cases}$ as desired. □

The next step is to express composition of strategies using the same language of polynomial functors. Remember from Section 2.1 that composition of strategies is standardly defined as follows. The composite $\sigma;\tau$ of two boolean strategies σ and τ over (A,B) and (B,C) respectively, is defined to accept all plays $p \in \mathbb{P}_{A,C}$ for which there exists $u \in \mathbb{P}_{A,B,C}$ such that $\delta_1(u) = p$ and $\delta_2(u)$ and $\delta_0(u)$ are accepted by σ and τ respectively. In [97], this is extended to a functor $\widehat{\mathbb{P}_{A,B}} \times \widehat{\mathbb{P}_{B,C}} \to \widehat{\mathbb{P}_{A,C}}$, whose definition is essentially a proof-relevant version of the boolean one:

Definition 6.2.10. *The composite $\sigma;\tau$ of two strategies σ and τ over (A,B) and (B,C) respectively, maps any play $p \in \mathbb{P}_{A,C}$ to the set of triples (u,x,y) where u is in $\mathbb{P}_{A,B,C}$ is such that $\delta_1(u) = p$, x is in $\sigma(\delta_2(u))$, and y in $\tau(\delta_0(u))$.*

Let us present this polynomially. First, by universal property of coproduct we have $\widehat{\mathbb{P}_{A,B}} \times \widehat{\mathbb{P}_{B,C}} \cong \widehat{\mathbb{P}_{A,B} + \mathbb{P}_{A,B}}$, so the problem reduces to defining a functor $\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \to \widehat{\mathbb{P}_{A,C}}$. Here is our candidate:

Definition 6.2.11. *Let $\mathsf{m}_{A,B,C}$ denote the polynomial functor*

$$\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \xrightarrow{\Delta_{\delta_2+\delta_0}} \widehat{\mathbb{P}_{A,B,C} + \mathbb{P}_{A,B,C}} \xrightarrow{\Pi_\nabla} \widehat{\mathbb{P}_{A,B,C}} \xrightarrow{\Sigma_{\delta_1}} \widehat{\mathbb{P}_{A,C}},$$

where $\nabla = [id, id]$.

This definition is legitimated by:

Proposition 6.2.12. *If* δ_1 *is a discrete fibration,* $\mathsf{m}_{A,B,C}$ *agrees with Definition 6.2.10, i.e., for all* σ *and* τ*, we have* $(\sigma;\tau) \cong \mathsf{m}_{A,B,C}[\sigma,\tau]$.

Proof. We can express left Kan extension along δ_1 as a sum by Lemma 2.2.27. We also have that ∇ is a discrete opfibration, so $\prod_\nabla(X)(u) \cong X(\mathrm{inl}\,u) \times X(\mathrm{inr}\,u)$ by Lemma 2.2.28, and therefore:

$$\begin{aligned}
\mathsf{m}_{A,B,C}[\sigma,\tau](p) &= \textstyle\sum_{\delta_1}\left(\prod_\nabla\left(\Delta_{\delta_2+\delta_0}[\sigma,\tau]\right)\right)(p)\\
&\cong \textstyle\sum_{\delta_1(u)=p}\prod_\nabla\left(\Delta_{\delta_2+\delta_0}[\sigma,\tau]\right)(u)\\
&\cong \textstyle\sum_{\delta_1(u)=p}\left(\Delta_{\delta_2+\delta_0}[\sigma,\tau](\mathrm{inl}\,u) \times \Delta_{\delta_2+\delta_0}[\sigma,\tau](\mathrm{inr}\,u)\right)\\
&\cong \textstyle\sum_{\delta_1(u)=p}[\sigma,\tau]((\delta_2+\delta_0)(\mathrm{inl}\,u)) \times [\sigma,\tau]((\delta_2+\delta_0)(\mathrm{inr}\,u))\\
&\cong \textstyle\sum_{\delta_1(u)=p}[\sigma,\tau](\mathrm{inl}(\delta_2(u))) \times [\sigma,\tau](\mathrm{inr}(\delta_0(u)))\\
&\cong \textstyle\sum_{\delta_1(u)=p}\sigma(\delta_2(u)) \times \tau(\delta_1(u)),
\end{aligned}$$

where $\sum$ means right Kan extension on the first line and disjoint union on all other lines, which clearly coincides with Tsukada and Ong's definition. □

Remark. *The discrete fibredness hypothesis is satisfied in most game models, with the notable exception of the saturated interpretation of AJM games (see Section 6.3.3), in which the projection is a non-discrete fibration.*

6.2.3 Game Settings, Associativity and Unitality

Required: 6.2.2, 6.1.2.
Recommended: ∅.

We have now expressed copycat strategies and composition abstractly, relying only on the postulated category-valued presheaf. Let us now consider associativity. It has become standard in game semantics to prove associativity of composition using a *zipping* result [9] stating that both squares

$$\begin{tikzcd}
\mathbb{P}_{A,B,C,D} \arrow[r, "\delta_2"] \arrow[d, "\delta_0"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{P}_{A,B,D} \arrow[d, "\delta_0"] & & \mathbb{P}_{A,B,C,D} \arrow[r, "\delta_1"] \arrow[d, "\delta_3"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{P}_{A,C,D} \arrow[d, "\delta_2"] \\
\mathbb{P}_{B,C,D} \arrow[r, "\delta_1"'] & \mathbb{P}_{B,D} & & \mathbb{P}_{A,B,C} \arrow[r, "\delta_1"'] & \mathbb{P}_{A,C}
\end{tikzcd} \tag{6.3}$$

are pullbacks. This holds in all considered game models.

Finally, we need to add one last bit to our axiomatisation to prove that copycat strategies are identities for composition. Suppose that u in $\mathbb{P}_{A,A,B}$ is such that $\delta_2(u) = \iota_0(s)$ for some sequence s in $\mathbb{P}_A$, then we intuitively want to have $u = \iota_0(\delta_0(u))$, which does not hold in general. We want both squares

$$\begin{tikzcd}
\mathbb{P}_{A,B} \arrow[r, "\delta_1"] \arrow[d, "\iota_0"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{P}_{A} \arrow[d, "\iota_0"] & & \mathbb{P}_{A,B} \arrow[r, "\delta_0"] \arrow[d, "\iota_1"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{P}_{B} \arrow[d, "\iota_0"] \\
\mathbb{P}_{A,A,B} \arrow[r, "\delta_2"'] & \mathbb{P}_{A,A} & & \mathbb{P}_{A,B,B} \arrow[r, "\delta_0"'] & \mathbb{P}_{B,B}
\end{tikzcd} \tag{6.4}$$

to be pullbacks, which is a slight generalisation of this intuition.

Definition 6.2.13. *A* game setting *consists of a set $\mathbb{A}$ (whose elements we call* games *or* arenas*) and a category-valued presheaf $\mathbb{P}$ on $\mathbb{A}_{[1,4]}$ such that all projections $\mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ and insertions $\mathbb{P}_A \to \mathbb{P}_{A,A}$ are discrete fibrations, and all squares* (6.3) *and* (6.4) *are pullbacks.*

We call both squares (6.3) the *zipping* squares of $\mathbb{P}$ and both squares (6.4) the *copycat* squares of $\mathbb{P}$.

Remark. *By zipping, because discrete fibrations are stable under pullbacks, all the horizontal projections in the squares* (6.3) *are discrete fibrations, and similarly, all insertions in the squares 6.4 also are.*

One of our main results is:

Theorem 6.2.14. *In any game setting, composition of strategies is associative up to isomorphism, and copycat strategies are units up to isomorphism.*

To prove this theorem, let us start with an alternative description of composition, which relies on the following intermediate category:

Definition 6.2.15. *For any triple of arenas A, B, C, let $\mathbb{P}_{(A,B),(B,C)}$ denote the* lax colimit *[61] of*

$$\mathbb{P}_{A,B} \xleftarrow{\delta_2} \mathbb{P}_{A,B,C} \xrightarrow{\delta_0} \mathbb{P}_{B,C},$$

i.e., the initial category equipped with natural transformations

$$\begin{tikzcd}
\mathbb{P}_{A,B} \arrow[ddrr, "l"'] & & \mathbb{P}_{A,B,C} \arrow[ll, "\delta_2"'] \arrow[rr, "\delta_0"] \arrow[dd, "m"] & & \mathbb{P}_{B,C} \arrow[ddll, "r"] \\
& \arrow[r, phantom, "\overset{\lambda}{\Longrightarrow}"] & & \arrow[l, phantom, "\overset{\rho}{\Longleftarrow}"] & \\
& & \mathbb{P}_{(A,B),(B,C)}. & &
\end{tikzcd}$$

The obtained category has as objects the disjoint union of objects from $\mathbb{P}_{A,B}$, $\mathbb{P}_{B,C}$, and $\mathbb{P}_{A,B,C}$. It inherits the corresponding morphisms, and has additional morphisms $\lambda_u : \delta_2 u \to u$ and $\rho_u : \delta_0 u \to u$ for all u satisfying the obvious naturality requirements that, for all $f : u \to u'$ in $\mathbb{P}_{A,B,C}$,

$$\begin{tikzcd}
\delta_2(u) \arrow[r, "\delta_2(f)"] \arrow[d, "\lambda_u"'] & \delta_2(u') \arrow[d, "\lambda_{u'}"] & & \delta_0(u) \arrow[r, "\delta_0(f)"] \arrow[d, "\rho_u"'] & \delta_0(u') \arrow[d, "\rho_{u'}"] \\
u \arrow[r, "f"'] & u' & & u \arrow[r, "f"'] & u'
\end{tikzcd} \tag{6.5}$$

commute. We have:

Proposition 6.2.16. *Composition is isomorphic to the polynomial functor*

$$\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \xrightarrow{\Pi_{[l,r]}} \widehat{\mathbb{P}_{(A,B),(B,C)}} \xrightarrow{\Delta_m} \widehat{\mathbb{P}_{A,B,C}} \xrightarrow{\Sigma_{\delta_1}} \widehat{\mathbb{P}_{A,C}}.$$

Proof. This follows from Lemma 2.2.65 by observing that the square

$$\begin{tikzcd}
\mathbb{P}_{A,B,C} + \mathbb{P}_{A,B,C} \arrow[rr, "\nabla"] \arrow[d, "\delta_2+\delta_0"'] & & \mathbb{P}_{A,B,C} \arrow[d, "m"] \\
\arrow[rr, phantom, "\Longrightarrow"] & & \\
\mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[rr, "{[l,r]}"'] & & \mathbb{P}_{(A,B),(B,C)}
\end{tikzcd}$$

is a cocomma square, where ∇ denotes the copairing $[id, id]$. Indeed, both categories have the same universal property, expressed differently. □

$$\begin{tikzcd}
\mathbb{P}_{A,B} + \mathbb{P}_{B,C} + \mathbb{P}_{C,D} \arrow[r, "\Pi"] \arrow[d, "\Pi"'] & \mathbb{P}_{(A,B),(B,C)} + \mathbb{P}_{C,D} \arrow[d, "\Pi"] & \mathbb{P}_{A,B,C} + \mathbb{P}_{C,D} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] & \mathbb{P}_{A,C} + \mathbb{P}_{C,D} \arrow[d, "\Pi"] \\
\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)} \arrow[r, "\Pi"'] & \mathbb{P}_{(A,B),(B,C),(C,D)} & & \mathbb{P}_{(A,C),(C,D)} \\
\mathbb{P}_{A,B} + \mathbb{P}_{B,C,D} \arrow[u, "\Delta"] \arrow[d, "\Sigma"'] & & \mathbb{P}_{A,B,C,D} \arrow[ul, "\Delta"'] \arrow[r, "\Sigma"] \arrow[d, "\Sigma"'] & \mathbb{P}_{A,C,D} \arrow[u, "\Delta"'] \arrow[d, "\Sigma"] \\
\mathbb{P}_{A,B} + \mathbb{P}_{B,D} \arrow[r, "\Pi"'] & \mathbb{P}_{(A,B),(B,D)} & \mathbb{P}_{A,B,D} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{P}_{A,D}
\end{tikzcd}$$

Figure 6.1: Diagram for associativity

Proving associativity thus reduces to showing that the perimeter of the diagram of Figure 6.1 commutes up to isomorphism. In order to do this, we introduce the category $\mathbb{P}_{(A,B),(B,C),(C,D)}$, similar to $\mathbb{P}_{(A,B),(B,C)}$ but with four arenas, which is constructed as the lax colimit of

$$\begin{tikzcd}
 & & \mathbb{P}_{B,C} & & \\
\mathbb{P}_{A,B} & \mathbb{P}_{A,B,C} \arrow[l, "\delta_2"'] \arrow[ur, "\delta_0"] & & \mathbb{P}_{B,C,D} \arrow[ul, "\delta_2"'] \arrow[r, "\delta_0"] & \mathbb{P}_{C,D} \\
 & & \mathbb{P}_{A,B,C,D}. \arrow[ul, "\delta_3"] \arrow[ur, "\delta_0"'] & &
\end{tikzcd}$$

It has as objects the objects of all six categories and morphisms as in those categories, plus morphisms $\delta_i(u) \to u$ whenever this makes sense (and composites thereof), with the same kind of naturality constraints as $\mathbb{P}_{(A,B),(B,C)}$.

In the diagram of Figure 6.1, both little squares commute up to isomorphism because the underlying squares do. It thus suffices to show that both heptagons commute up to isomorphism. Both cases are symmetric, so we only treat the bottom left one. We then need to introduce yet another category, $\mathbb{P}_{(A,B),(B,C,D)}$, which is like $\mathbb{P}_{(A,B),(B,C),(C,D)}$, except that $\mathbb{P}_{B,C,D}$ is not decomposed into $\mathbb{P}_{B,C}$ and $\mathbb{P}_{C,D}$: it is the lax colimit of

$$\mathbb{P}_{A,B} \xleftarrow{\delta_2\delta_2} \mathbb{P}_{A,B,C,D} \xrightarrow{\delta_0} \mathbb{P}_{B,C,D}.$$

These categories are related by the full embeddings given by the universal properties of $\mathbb{P}_{(A,B),(B,C)}$ and $\mathbb{P}_{(A,B),(B,C,D)}$ (indeed, $\mathbb{P}_{(A,B),(B,C),(C,D)}$ has lax cocones from $\mathbb{P}_{A,B} \leftarrow \mathbb{P}_{A,B,C} \to \mathbb{P}_{B,C}$ and $\mathbb{P}_{A,B} \leftarrow \mathbb{P}_{A,B,C,D} \to \mathbb{P}_{B,C,D}$ given by its own definition):

$$\mathbb{P}_{(A,B),(B,C)} \hookrightarrow \mathbb{P}_{(A,B),(B,C),(C,D)} \hookleftarrow \mathbb{P}_{(A,B),(B,C,D)}.$$

The crucial reason why the heptagon commutes is:

Lemma 6.2.17. *The square*

$$\begin{tikzcd}
\mathbb{P}_{A,B} + \mathbb{P}_{B,C,D} \arrow[r, "{[l,r]}"] \arrow[d, "\mathbb{P}_{A,B}+\delta_1"'] & \mathbb{P}_{(A,B),(B,C,D)} \arrow[d, "\mathbb{P}_{(A,B),\delta_1}"] \\
\mathbb{P}_{A,B} + \mathbb{P}_{B,D} \arrow[r, "{[l,r]}"'] & \mathbb{P}_{(A,B),(B,D)}
\end{tikzcd}$$

is a local pushforward square.

Proof. The square is obviously a pullback, so, by Lemma 2.2.84, it suffices to show that $\partial^*(\mathbb{P}_{(A,B),\delta_1})$ is a sheaf for the Grothendieck topology induced by the embedding $[l,r]$.

Remember that, if $F{:}\,\mathbb{C} \to \mathbb{D}$ is a functor, a sieve S covers an object d of $\mathbb{D}$ for the topology induced by F if and only if it contains all morphisms of the form $Fc \to d$. In our case, a sieve S covers $l(p_{A,B})$ if and only if it contains all morphisms of the form $[l,r](p) \to l(p_{A,B})$ for p in $\mathbb{P}_{A,B} + \mathbb{P}_{B,D}$, but since there are no morphisms of the form $r(-) \to l(-)$, S covers $l(p_{A,B})$ if and only if it contains all morphisms of the form $l(p) \to l(p_{A,B})$ for p in $\mathbb{P}_{A,B}$. In particular, it must contain $id_{l(p_{A,B})} = l(id_{p_{A,B}}){:}\,l(p_{A,B}) \to l(p_{A,B})$, so it must be the maximal sieve. Similarly, a sieve S covers $r(p_{B,D})$ if and only if it is the maximal sieve. The sheaf condition is therefore not constraining on objects of the form $l(p_{A,B})$ and $r(p_{B,D})$ (all matching families necessarily have a unique amalgamation given by the identity morphism). For objects of the form $m(u_{A,B,D})$, a sieve S is covering if and only if it contains all morphisms of the form $[l,r](p) \to m(u_{A,B,D})$, but since all such morphisms can be factored as either $\lambda_{u_{A,B,D}} f$ or $\rho_{u_{A,B,D}} f$, a sieve is covering if and only if it contains $\lambda_{u_{A,B,D}}$ and $\rho_{u_{A,B,D}}$.

Consider a sieve S covering $u_{A,B,D}$. A matching family for S is a family $x_f \in \partial^*(\mathbb{P}_{(A,B),\delta_1})(c)$ for all morphisms $f{:}\,c \to m(u_{A,B,D})$ in S such that $x_f \cdot g = x_{fg}$ for all f and g, and an amalgamation for such a family is an $x \in \partial^*(\mathbb{P}_{(A,B),\delta_1})(u_{A,B,D})$ such that $x \cdot f = x_f$ for all $f \in S$. If we consider any matching family x_- for S, since $\partial^*(X)(c) \cong X^{-1}(c)$ and S contains $\rho_{u_{A,B,D}}$, $x_{\rho_{u_{A,B,D}}}$ is an element of $\mathbb{P}^{-1}_{(A,B),\delta_1}(r(\delta_0(u_{A,B,D}))) \cong \delta_1^{-1}(\delta_0(u_{A,B,D}))$, i.e., an element $u_{B,C,D}$ of $\mathbb{P}_{B,C,D}$ such that $\delta_1(u_{B,C,D}) = \delta_0(u_{A,B,D})$. Therefore, by zipping, we get an element $u_{A,B,C,D}$ of $\mathbb{P}_{A,B,C,D}$ that projects to $u_{A,B,D}$ and $u_{B,C,D}$, which is exactly an element of $\mathbb{P}^{-1}_{(A,B),\delta_1}(m(u_{A,B,D}))$. We take $u_{A,B,C,D}$ as our candidate amalgamation x. It is the only possible choice of amalgamation, because, any amalgamation $u'_{A,B,C,D}$ must project to $u_{A,B,D}$ (because it must be an element over $m(u_{A,B,D})$) and also to $u_{B,C,D}$ (because we must have $x \cdot \rho_{u_{A,B,D}} = x_{\rho_{u_{A,B,D}}} = u_{B,C,D}$), so it must be equal to $u_{A,B,C,D}$ by the zipping squares (6.3) being pullbacks.

We now show that it is indeed an amalgamation, i.e., that $x \cdot f = x_f$ for all $f \in S$. This is obviously the case for objects of the form $l(p_{A,B})$ because there is only one element above each such object. For objects of the form $r(p_{B,D})$, the result come from the fact that all morphisms $r(p_{B,D}) \to m(u_{A,B,D})$ factors uniquely as $r(p_{B,D}) \xrightarrow{r(f)} r(\delta_0(u_{A,B,D})) \xrightarrow{\rho_{u_{A,B,D}}} m(u_{A,B,D})$, and we have chosen x such that $x \cdot \rho_{u_{A,B,D}} = u_{B,C,D}$. For objects of the form $m(u'_{A,B,D})$, the result comes from the fact that elements above $m(u'_{A,B,D})$ are in bijection with those above $r(\delta_0(u'_{A,B,D}))$ through $- \cdot \rho_{u'_{A,B,D}}$, and that the result holds for those objects. □

$$\begin{tikzcd}
\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)} \arrow[dd, "\Pi"'] & \mathbb{P}_{A,B} + \mathbb{P}_{B,C,D} \arrow[l, "\Delta"'] \arrow[rr, "\Sigma"] \arrow[d, "\Pi"'] \arrow[dr, "\Pi"] & & \mathbb{P}_{A,B} + \mathbb{P}_{B,D} \arrow[d, "\Pi"] \\
& \mathbb{P}_{(A,B),(B,C),(C,D)} & \mathbb{P}_{(A,B),(B,C,D)} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] & \mathbb{P}_{(A,B),(B,D)} \\
\mathbb{P}_{(A,B),(B,C),(C,D)} & \mathbb{P}_{A,B,C,D} \arrow[l, "\Delta"] \arrow[rr, "\Sigma"'] \arrow[u, "\Delta"] \arrow[ur, "\Delta"'] & & \mathbb{P}_{A,B,D} \arrow[u, "\Delta"']
\end{tikzcd}$$

Figure 6.2: Zoom into the bottom left heptagon of Figure 6.1

This leads us to fill the heptagon in Figure 6.2. The top right square commutes by Lemmas 6.1.2 and 6.2.17. The top triangle commutes up to isomorphism by Lemma 2.2.68, the bottom one because the underlying diagram commutes, and the bottom-right square by Lemma 2.2.70.

Associativity finally follows from:

Lemma 6.2.18. *The left square commutes up to isomorphism.*

Proof. The classical limit formula for right Kan extensions gives that the right Kan extension of a presheaf X over $\mathbb{C}$ along $F{:}\,\mathbb{C} \to \mathbb{D}$ is given at d by the limit of $(\mathbb{C} \downarrow d)^{op} \xrightarrow{\partial_d} \mathbb{C}^{op} \xrightarrow{X} \mathsf{Set}$. So, in our case:

- following the left and bottom arrows we obtain a presheaf mapping any $u \in \mathbb{P}_{A,B,C,D}$ to the limit of
$$((\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)}) \downarrow u)^{op} \to (\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)})^{op} \to \mathsf{Set},$$
- following the top and right arrows, we obtain a presheaf mapping any u to the limit of
$$((\mathbb{P}_{A,B} + \mathbb{P}_{B,C,D}) \downarrow u)^{op} \to (\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)})^{op} \to \mathsf{Set},$$

where u is seen as an object of $\mathbb{P}_{(A,B),(B,C),(C,D)}$ in the formulas. Now, the inclusion functor $(\mathbb{P}_{A,B} + \mathbb{P}_{B,C,D}) \downarrow u \to (\mathbb{P}_{A,B} + \mathbb{P}_{(B,C),(C,D)}) \downarrow u$ is readily checked to be final [76, IX.3], so its opposite is initial and both limits are isomorphic. □

This ends the proof of associativity.

Left and right unitality are entirely symmetric, so we only treat one. First, we observe that, because $\iota_0{:}\,\mathbb{P}_A \to \mathbb{P}_{A,A}$ is a discrete fibration, so is $\iota_0 + \mathbb{P}_{A,B}{:}\,\mathbb{P}_A + \mathbb{P}_{A,B} \to \mathbb{P}_{A,A} + \mathbb{P}_{A,B}$. In the diagram of Figure 6.3, the top path is composition of a strategy on $\mathbb{P}_{A,B}$ with the copycat strategy on $\mathbb{P}_{A,A}$ and its bottom path is isomorphic to the identity (because the first three morphisms are isomorphic to the identity and $\delta_1 \iota_0 = \mathbb{P}(\mathsf{d}^2_1)\mathbb{P}(\mathsf{i}^2_0) = \mathbb{P}(\mathsf{i}^2_0 \mathsf{d}^2_1) = \mathbb{P}(id_{[2]}) = id_{\mathbb{P}_{A,B}}$). So if we manage to show that all the squares commute, we will have proven (left) unitality.

To pave the diagram, we introduce the category $\mathbb{P}_{A,(A,B)}$, which we define as the lax colimit of

$$\begin{tikzcd}
\mathbb{P}_A & \mathbb{P}_{A,B} \arrow[l, "\delta_1"'] \arrow[r, equal] & \mathbb{P}_{A,B}
\end{tikzcd}$$

$$\begin{tikzcd}
\varnothing + \mathbb{P}_{A,B} \arrow[r, "\Pi_{!+\mathbb{P}_{A,B}}"] \arrow[d, "\Pi_{[!,\mathbb{P}_{A,B}]}"'] & \mathbb{P}_A + \mathbb{P}_{A,B} \arrow[r, "\Sigma_{\iota_0+\mathbb{P}_{A,B}}"] \arrow[d, "\Pi_{[l,r]}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{P}_{A,A} + \mathbb{P}_{A,B} \arrow[d, "\Pi_{[l,r]}"] & \\
\mathbb{P}_{A,B} \arrow[r, "\Pi_r"] \arrow[d, equal] \arrow[dr, phantom, "\overset{\rho}{\Longrightarrow}"] & \mathbb{P}_{A,(A,B)} \arrow[r, "\Sigma_{\mathbb{P}_{\iota_0,(A,B)}}"] & \mathbb{P}_{(A,A),(A,B)} & \\
\mathbb{P}_{A,B} \arrow[r, equal] & \mathbb{P}_{A,B} \arrow[u, "\Delta_m"] \arrow[r, "\Sigma_{\iota_0}"'] \arrow[ur, phantom, "\urcorner", very near start] & \mathbb{P}_{A,A,B} \arrow[u, "\Delta_m"'] \arrow[r, "\Sigma_{\delta_1}"'] & \mathbb{P}_{A,B}.
\end{tikzcd}$$

Figure 6.3: Diagram for left unitality

and $\mathbb{P}_{\iota_0,(A,B)}: \mathbb{P}_{A,(A,B)} \to \mathbb{P}_{(A,A),(A,B)}$ by its universal property. The top-left square commutes up to isomorphism because the underlying diagram does, the bottom-right because it is exact by Lemma 2.2.71.

The bottom-left square commutes as well because the underlying square is exact, which we prove by the zigzag criterion: by construction of $\mathbb{P}_{A,(A,B)}$, any morphism $r(p) \to m(p')$ in $\mathbb{P}_{A,(A,B)}$ can be factored as $r(p) \xrightarrow{r(f)} r(p') \xrightarrow{\rho_{p'}} m(p')$, which is our candidate factorisation. For any other such factorisation $r(p) \xrightarrow{r(g)} r(q) \xrightarrow{\rho_q} m(q) \xrightarrow{m(h)} m(p')$, we have by construction of $\mathbb{P}_{A,(A,B)}$ that $hg = f$, from which we get that

$$\begin{tikzcd}
 & p \arrow[dl, "g"'] \arrow[dr, "f"] & \\
q \arrow[rr, "h"'] & & p' \\
q \arrow[rr, "h"] \arrow[dr, "h"'] & & p' \\
 & p' \arrow[ur, equal] &
\end{tikzcd}$$

is a lantern.

Finally, the crucial reason why the diagram commutes is:

Lemma 6.2.19. *The square*

$$\begin{tikzcd}
\mathbb{P}_A + \mathbb{P}_{A,B} \arrow[r, "{[l,r]}"] \arrow[d, "\iota_0+\mathbb{P}_{A,B}"'] & \mathbb{P}_{A,(A,B)} \arrow[d, "\mathbb{P}_{\iota_0,(A,B)}"] \\
\mathbb{P}_{A,A} + \mathbb{P}_{A,B} \arrow[r, "{[l,r]}"'] & \mathbb{P}_{(A,A),(A,B)}
\end{tikzcd}$$

is a local pushforward square.

Proof. First, the square is obviously a pullback. It is also routine to check that $\mathbb{P}_{\iota_0,(A,B)}$ is a discrete fibration. So, by Lemma 2.2.84, it suffices to show that $\partial^*(\mathbb{P}_{\iota_0,(A,B)})$ is a sheaf. Given a sieve S covering an object of the form $l(p)$ or $r(p)$, it is trivial to show there is a unique amalgamation for any compatible family for S, because S necessarily contains the identity. For objects of the form $m(u)$, a sieve is covering if and only if it contains λ_u and ρ_u. In particular, a compatible family x_- for a sieve S covering u must contain an element $s = x_{\lambda_u}$ over $r(\delta_2(u))$. This element is such that $\iota_0(s) = \delta_2(u)$ (because it is above $r(\delta_2(u))$ and that elements above $r(p)$ are antecedents of p through ι_0). Therefore, because the copycat squares (6.4) are pullbacks, we get an element p such that $\delta_1(p) = s$ and $\iota_0(p) = u$. In particular, it is an element above u, which we take as our candidate amalgamation x. This is the only possible choice of amalgamation because an amalgamation x must be an element over u (which implies $\iota_0(x) = u$) and must restrict to the compatible family, in particular $x \cdot \lambda_u = x_{\lambda_u} = \delta_2(u)$, and the copycat squares are pullbacks, so there is a unique such element.

We now show that p is indeed an amalgamation. It restricts to the desired elements over objects of the form $r(p)$ because there is only one element above such an object. For elements of the form $l(p)$, the result comes from the fact that all morphisms $l(p) \to m(u)$ can be written as $l(p) \xrightarrow{l(f)} l(\delta_2(u)) \xrightarrow{\lambda_u} m(u)$ and p admitting the desired restriction to x_{λ_u} by construction. For elements of the form $m(u')$, the result comes from the fact elements above are in bijection with those above $l(\delta_2(u'))$ through $- \cdot \lambda_{u'}$, and the result holds for those objects. □

This ends the proof of (left) unitality.

6.2.4 The Boolean Case

Required: 6.2.3.
Recommended: ∅.

Let us conclude this section by treating the boolean case: until now, our strategies were given by general presheaves (Definition 6.2.7). We would like to derive from Theorem 6.2.14 that boolean strategies also form a category.

The bridge to the boolean case is given by the embedding $\mathsf{r} \colon 2 \hookrightarrow \mathsf{Set}$ mapping $0 \leq 1$ to $\varnothing \to 1$. This functor has a left adjoint l mapping $\varnothing$ to 0 and collapsing all non-empty sets to 1. Furthermore, r being fully faithful, we have in fact a full reflection, which induces a further one between presheaves and boolean presheaves:

Proposition 6.2.20. *For any small category $\mathbb{C}$, post-composition by l and r yield a full reflection*

$$\begin{tikzcd}
{[\mathbb{C}^{op}, \mathsf{Set}]} \arrow[r, bend left, "\mathsf{l}_!"] \arrow[r, phantom, "\perp"] & {[\mathbb{C}^{op}, 2]} \arrow[l, bend left, "\mathsf{r}_!"]
\end{tikzcd}.$$

The left adjoint $\mathsf{l}_!$ is called booleanisation*.*

Proof. The pair of functors $\mathsf{l}_! \dashv \mathsf{r}_!$ clearly forms an adjunction and $\mathsf{r}_!$ is clearly a full embedding. □

Because 2 is complete and cocomplete, replacing Set with 2 in Definition 6.2.8 yields a notion of *boolean* polynomial functor:

Notation 6.2.21. *Any functor $F \colon \mathbb{C} \to \mathbb{D}$ induces restriction, left Kan extension and right Kan extension functors between boolean presheaf categories $\widetilde{\mathbb{C}}$ and $\widetilde{\mathbb{D}}$, respectively denoted by $\overline{\Delta_F}$, $\overline{\sum_F}$ and $\overline{\prod_F}$. Accordingly, the boolean version of any polynomial functor P will be denoted by $\overline{P}$.*

We may thus transfer our polynomial definitions of copycat and composition to boolean strategies. Concrete examples of game settings will be considered in Section 6.3, for which we have:

Proposition 6.2.22. *In all the game settings of Section 6.3,* $\overline{\mathsf{m}}$ *coincides with standard composition.*

To show that our results in the presheaf case transfer to the boolean case, we show that the polynomial functors we used commute with booleanisation. This is easy for left extensions and restrictions:

Proposition 6.2.23. *For all functors $F{:}\,\mathbb{C} \to \mathbb{D}$, the following squares commute up to isomorphism.*

$$\begin{tikzcd}
{[\mathbb{C}^{op}, \mathsf{Set}]} \arrow[r, "\Sigma_F"] \arrow[d, "\mathsf{l}_!"'] & {[\mathbb{D}^{op}, \mathsf{Set}]} \arrow[d, "\mathsf{l}_!"] \\
{[\mathbb{C}^{op}, 2]} \arrow[r, "\bar{\Sigma}_F"'] & {[\mathbb{D}^{op}, 2]}
\end{tikzcd}
\qquad
\begin{tikzcd}
{[\mathbb{C}^{op}, \mathsf{Set}]} \arrow[d, "\mathsf{l}_!"'] & {[\mathbb{D}^{op}, \mathsf{Set}]} \arrow[l, "\Delta_F"'] \arrow[d, "\mathsf{l}_!"] \\
{[\mathbb{C}^{op}, 2]} & {[\mathbb{D}^{op}, 2]} \arrow[l, "\bar{\Delta}_F"]
\end{tikzcd}$$

Proof. Commutation with restriction is obvious because $\mathsf{l}_!$ is post-composition by l and Δ_F is pre-composition by F. For left Kan extension, the reflection l, being a left adjoint, preserves colimits, which is precisely what $\sum_F$ computes. □

Things do not work out so well with right extensions in general. In order to show that our polynomial functors commute with booleanisation, it is thus useful to delineate a sufficiently large class of limits that are preserved by l:

Lemma 6.2.24. *The left adjoint l preserves products.*

Proof. A product $\prod_i X_i$ is non-empty just when each X_i is, hence just when $\mathsf{l}(X_i) = 1$ for all i, i.e., when $\prod_i \mathsf{l}(X_i) = 1$. Thus $\mathsf{l}(\prod_i X_i) = 1$ if and only if $\prod_i \mathsf{l}(X_i) = 1$, hence $\mathsf{l}(\prod_i X_i) = \prod_i \mathsf{l}(X_i)$. □

Proposition 6.2.25. *booleanisation commutes with $\prod_F$, for any $F{:}\,\mathbb{C} \to \mathbb{D}$ such that for all $d \in \mathrm{ob}(\mathbb{D})$ the comma category F/d is a coproduct of categories with a terminal object.*

Proof. Indeed, consider any such F. For any $X \in \widehat{\mathbb{C}}$ and $d \in \mathbb{D}$, letting $F/d \cong \sum_{i \in n_d} \mathbb{D}^d_i$ with $\varphi^d_i{:}\,F(c^d_i) \to d$ denoting the terminal object in $\mathbb{D}^d_i$, we have that $[[F-,d],X] \cong \prod_{i \in n_d} X(c^d_i)$. Indeed, we may map any natural transformation α to the tuple $(\alpha_{c^d_i}(\varphi^d_i))_{i \in n_d}$ and conversely any tuple $(x_i)_{i \in n_d}$ to the transformation that maps $f{:}\,Fc \to d$ to $x_i \cdot g$, where $f = \varphi^d_i F(g)$ is given by universal property of terminal object. These maps are inverse to each other because α is natural. Therefore, we have:

$$\begin{aligned}
\mathsf{l}_!(\textstyle\prod_F(X))(d) &= \mathsf{l}(\textstyle\prod_F(X)(d)) \\
&\cong \mathsf{l}([[F-,d],X]) \\
&\cong \mathsf{l}(\textstyle\prod_{i \in n_d} X(c^d_i)) \\
&\cong \textstyle\prod_{i \in n_d} \mathsf{l}(X(c^d_i)) && \text{(by Lemma 6.2.24)} \\
&\cong \textstyle\prod_{i \in n_d} \mathsf{l}_!(X)(c^d_i) \\
&\cong \textstyle\prod_{i \in n_d} [[F-,d],\mathsf{l}_!(X)] \\
&\cong \textstyle\prod_F(\mathsf{l}_!(X))(d),
\end{aligned}$$

as desired. □

Proposition 6.2.26. *The class of functors F such that $\prod_F$ commutes with booleanisation contains all functors $\nabla_{\mathbb{C}} : \mathbb{C} + \mathbb{C} \to \mathbb{C}$ and $! : \varnothing \to \mathbb{C}$, and it is closed under composition and coproduct (i.e., $F + G : \mathbb{C} + \mathbb{C}' \to \mathbb{D} + \mathbb{D}'$ is in it if F and G are).*

Proof. This is direct for composition and easy consequences of the previous proposition for everything else. □

As desired, we obtain:

Proposition 6.2.27. *In any game setting, composition of boolean strategies is associative and unital up to isomorphism.*

Proof. Both results state that two polynomial functors, say $\overline{P_1}$ and $\overline{P_2}$ between categories of the form $[\mathbb{C}^{op}, 2]$ are naturally isomorphic. But knowing that their set-versions, say P_1 and P_2, are isomorphic, we may form

$$\begin{tikzcd}
{[\mathbb{C}^{op}, \mathsf{Set}]} \arrow[r, bend left, "P_1"] \arrow[r, bend right, "P_2"'] \arrow[d, "\mathsf{l}_!"'] & {[\mathbb{D}^{op}, \mathsf{Set}]} \arrow[d, "\mathsf{l}_!"] \\
{[\mathbb{C}^{op}, 2]} \arrow[r, bend left, "\overline{P_1}"] \arrow[r, bend right, "\overline{P_2}"'] & {[\mathbb{D}^{op}, 2].}
\end{tikzcd}$$

In both cases, this diagram commutes serially by Propositions 6.2.23 and 6.2.26, and we have proved that the top parallel functors are isomorphic. But $\mathsf{l}_!$ is epi, which entails that the bottom parallel functors are also isomorphic, as desired. □

Remark. *Please note that we have not claimed that boolean composition agrees with general, set-based composition, i.e., commutation of the diagram below.*

$$\begin{tikzcd}
\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\mathsf{m}"] & \widehat{\mathbb{P}_{A,C}} \\
\widetilde{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\overline{\mathsf{m}}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}} \arrow[u, "\mathsf{r}_!"']
\end{tikzcd}$$

In fact it does not in general, and this is the main cause for the failure of stability of boolean, innocent strategies under composition [46, Section 3.7.2]. What does hold, however, is

- *commutation of booleanisation with composition as on the left below,*
- *the characterisation of boolean composition given below right, as set-based composition followed by booleanisation (because $\mathsf{l}_! \dashv \mathsf{r}_!$ is a full reflection, so the counit is an isomorphism).*

$$\begin{tikzcd}
\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\mathsf{m}"] \arrow[d, "\mathsf{l}_!"'] & \widehat{\mathbb{P}_{A,C}} \arrow[d, "\mathsf{l}_!"] \\
\widetilde{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\overline{\mathsf{m}}"'] & \widetilde{\mathbb{P}_{A,C}}
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\mathsf{m}"] & \widehat{\mathbb{P}_{A,C}} \arrow[d, "\mathsf{l}_!"] \\
\widetilde{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \arrow[r, "\overline{\mathsf{m}}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}}
\end{tikzcd}$$

Let us move on to exhibit a few concrete game settings. We will return to the boolean case in Section 6.4.3, to deal with innocence.

6.3 Applications

In this section, we show that a number of standard game models fit into our framework. In Section 6.3.1, we consider HON games, in their p-form, which by the results of Section 6.2 yields categories of pg and pb-strategies. We then refine our results by considering variants in which some constraints are imposed on strategies (or equivalently plays) in Section 6.3.2: first a local form of constraint, followed by a slightly more involved form, obtained by enriching games with validity predicates on plays. These variants are shown to form game settings (hence yield categories of pg and pb-strategies). AJM games are considered in Section 6.3.3, and also shown to form a game setting. Finally, we explain in Section 6.3.4 why Blass games fail to form a game setting.

6.3.1 Hyland-Ong/Nickau Games

Required: 6.2.4, 2.1.1. **Recommended:** ∅.

We define arenas, plays, interaction sequences, etc, as in Section 2.1.1.

A classical lemma in HON games is (and can for example be found in [96]):

Proposition 6.3.1. *A justified sequence s on an arena triple (A, B, C) is an interaction sequence if and only if its projection to (A, B), (B, C), and (A, C) are plays.*

Therefore, we have the wanted projection δ_i from interaction sequences to plays. Furthermore, a similar lemma holds for generalised interaction sequences, yielding projections from generalised interaction sequences to interaction sequences.

Let us finally define the insertion functors. The intuition is simply to create two copies of the kth arena and play whatever is played on it in both copies (Opponent moves are played in the right-hand copy first, while Proponent moves are played in the left-hand copy first, which mimics program flow).

Let us start with the simplest case, where s is in $\mathbb{P}_A$.

Definition 6.3.2. *If $s = (n, f, \varphi)$ is a justified sequence on A, then we define $\iota_0(s)$ to be the sequence $\tilde{s} = (2n, \tilde{f}, \tilde{\varphi})$ with $\tilde{f}(2i - 1) = f(i)_{a_i}$ and $\tilde{f}(2i) = f(i)_{\overline{a_i}}$ (where a_i denotes the right-hand copy of A if $f(i)$ is an Opponent move and the other copy otherwise, and $\overline{a_i}$ is the copy of A that is not a_i), $\tilde{\varphi}(2i - 1) = 2\varphi(i)$, and*

$$\tilde{\varphi}(2i) = \begin{cases} 2i - 1 & \textit{if } f(i) \textit{ is an initial move} \\ 2\varphi(i) - 1 & \textit{otherwise.} \end{cases}$$

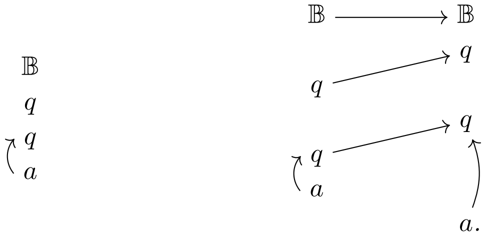
Proposition 6.3.3. *The mapping ι_0 thus defined is a functor from $\mathbb{P}_A$ to $\mathbb{P}_{A,A}$.*

Proof. We want to show that, for any justified sequence s in $\mathbb{P}_A$, $\tilde{s}$ is an alternating justified sequence of even length, and everything is direct except for the fact that it is justified. Then $\tilde{f}(\tilde{\varphi}(2i-1)) = \tilde{f}(2\varphi(i-1)) = f(\varphi(i-1))_{\overline{a_{\varphi(i-1)}}}$. Because

s is a justified sequence, $f(\varphi(i-1))$ and $f(i-1)$ have opposite polarities, so $\overline{a_{\varphi(i-1)}} = a_{i-1}$, so $\tilde{f}(\tilde{\varphi}(2i-1)) = f(\varphi(i-1))_{a_{i-1}} \vdash f(i-1)_{a_{i-1}} = \tilde{f}(2i-1)$. If $f(i)$ is initial, we have $\tilde{f}(\tilde{\varphi}(2i)) = \tilde{f}(2i-1) = f(i)_r \vdash f(i)_l = \tilde{f}(2i)$. And finally, when $f(i)$ is not initial, $\tilde{f}(2i) = \tilde{f}(2\varphi(i)-1) = f(\varphi(i))_{a_{\varphi(i)}} \vdash f(i)_{a_{\varphi(i)}} = f(i)_{\overline{a_i}} = \tilde{f}(2i)$.

Finally, to show that ι_0 is a functor, we need to show that, if s is a prefix of t, then $\tilde{s}$ is a prefix of $\tilde{t}$, which is direct. □

Example 6.3.4. *The justified sequence on $\mathbb{B}$ below left, which is not alternating, yields the copycat play on the right:*



Note that this example shows that it is indeed crucial to take $\mathbb{P}_A$ to be the set of all justified sequences in our construction if we want to get all copycat plays in the image of $\varnothing \xrightarrow{\Pi} \mathbb{P}_A \xrightarrow{\Sigma} \mathbb{P}_{A,A}$.

In general, the insertion functor $\iota_k : \mathbb{P}_L \to \mathbb{P}_{L+k}$ maps the justified sequence $s = (n, f, \varphi)$ to $\tilde{s} = (n + n_k, \tilde{f}, \tilde{\varphi})$ (where n_k is the number of moves on the kth arena in s) defined as follows.

We first define a surjective map $\alpha : n + n_k \to n$ that maps each occurrence of a move in $\tilde{s}$ to the move of s it copies: if $f(i)$ is on the kth arena, then i has two antecedents through α, and only one otherwise. It is defined as $\alpha(0) = 0$ (for the induction to go through), $\alpha(1) = 1$, and

$$\alpha(i+1) = \begin{cases} \alpha(i) & \text{if } f(\alpha(i)) \text{ is in the } k\text{th arena and } \alpha(i) \neq \alpha(i-1) \\ \alpha(i)+1 & \text{otherwise.} \end{cases}$$

We can now define $\tilde{f}$ and $\tilde{\varphi}$ using this map. Let us denote by $\overline{f(i)}$ the index of the arena $f(i)$ is played in (in s) and by m_i the move m if it is played in the ith arena (in $\tilde{s}$), then we define $\tilde{f}$ as:

$$\tilde{f}(i) = \begin{cases} f(\alpha(i))_{\mathsf{d}_k(j)} & \text{if } \overline{f(\alpha(i))} = j \neq k \\ f(\alpha(i))_{k+1} & \text{if } \alpha(i) \neq \alpha(i-1) \text{ and } f(\alpha(i)) \text{ is an Opponent move} \\ & \text{or } \alpha(i) = \alpha(i-1) \text{ and } f(\alpha(i)) \text{ is a Proponent move} \\ f(\alpha(i))_k & \text{otherwise,} \end{cases}$$

where $\mathsf{d}_k : p \to p+1$ misses k: $\mathsf{d}_k(i) = i$ if $i < k$ and $\mathsf{d}_k(i) = i+1$ if $i \geq k$ (where p is the number of arenas s is played on).

For $\tilde{\varphi}$, things are slightly messier because the initial moves in the kth arena should now be justified by their predecessors, but the rest of the pointers should be given by s. To define it, let us notice that all i such that $f(i)$ is in the kth arena have two antecedents through α, say i_1 and i_2, and that one of them is such that $\tilde{f}(i_1)$ is in the kth arena and the other is in the $k+1$th arena. Let us call $\alpha_k^{-1}(i)$ and $\alpha_{k+1}^{-1}(i)$ these two antecedents. If we generalise this notation, we get $\alpha_j^{-1}(i)$, which denotes the antecedent of i that is played in the jth arena. Two such antecedents are always equal, but they may not exist. However, there is always a j such that i has an antecedent through α_j, and it has two such antecedents if and only if $\overline{f(i)} = k$ (in which case its two antecedents are the ones described above).

Lemma 6.3.5. *If $s = (n, f, \varphi)$, then for all i in n, if $\alpha_j^{-1}(i)$ exists, then $\tilde{f}(\alpha_j^{-1}(i)) = f(i)_j$.*

Proof. Direct. □

We define $\tilde{\varphi}$ by:

$$\tilde{\varphi}(\alpha_j^{-1}(i)) = \begin{cases} \alpha_{j+1}^{-1}(i) & \text{if } j = k \text{ and } f(i) \text{ is initial in } A_j \\ \alpha_{j+1}^{-1}(\varphi(i)) & \text{if } f(i) \text{ is initial in } A_j \\ \alpha_j^{-1}(\varphi(i)) & \text{otherwise,} \end{cases}$$

where, by convention, $\alpha_{p+1}^{-1}(0) = 0$ if s is played on p arenas. (The reader may notice that this construction is a generalisation of the construction of $\iota_0 \colon \mathbb{P}_A \to \mathbb{P}_{A,A}$ defined above.)

Proposition 6.3.6. *If s is a play (resp. an interaction sequence), then $\tilde{s}$ is an interaction sequence (resp. a generalised interaction sequence). Furthermore, the mappings thus defined extend to functors.*

Proof. On objects, it is enough to show that $\tilde{s}$ is a justified sequence that ends in an outmost arena and whose first and last projections are plays (resp. interaction sequences).

The easiest point is that $\tilde{s}$ ends in one of the outmost arenas. This is direct if k is not one of the outmost arenas or if it is one of them, but s ends in the other outmost arena. Otherwise $\tilde{f}(n + n_k) = f(\alpha(n + n_k))_{k+\varepsilon} = f(n)_{k+\varepsilon}$, where ε equals 1 if and only if $f(n)$ is a Proponent move (because we know that $\alpha(n + n_k) = \alpha(n + n_k - 1)$). But we know that s must end with a Proponent move if it ends in its rightmost arena and with an Opponent move otherwise, so we have that

$$\tilde{f}(n + n_k) = \begin{cases} f(n)_{1+0} & \text{if } s \text{ ends in its leftmost arena} \\ f(n)_{p+1} & \text{if } s \text{ ends in its rightmost arena } A_p. \end{cases}$$

Let us now show that $\tilde{s}$ is justified. We consider any $\alpha_j^{-1}(i)$ for some i in n and show that $\tilde{f}(\tilde{\varphi}(\alpha_j^{-1}(i))) \vdash \tilde{f}(\alpha_j^{-1}(i))$.

- The simplest case is when $f(i)$ is not initial in A_j, in which case we have $\tilde{f}(\tilde{\varphi}(\alpha_j^{-1}(i))) = \tilde{f}(\alpha_j^{-1}(\varphi(i))) = f(\varphi(i))_j \vdash f(i)_j = \tilde{f}(\alpha_j^{-1}(i))$, using Lemma 6.3.5.
- Now, if $f(i)$ is initial in A_j, there are two cases.
  - If $j = k$, then we have $\tilde{f}(\tilde{\varphi}(\alpha_j^{-1}(i))) = \tilde{f}(\alpha_{j+1}^{-1}(i)) = f(i)_{j+1} \vdash f(i)_j = \tilde{f}(\alpha_j^{-1}(i))$.
  - Otherwise, $\tilde{f}(\tilde{\varphi}(\alpha_j^{-1}(i))) = \tilde{f}(\alpha_{j+1}^{-1}(\varphi(i)))) = f(\varphi(i))_{j+1} \vdash f(i)_j = \tilde{f}(\alpha_j^{-1}(i))$.

The last thing to prove on objects is that the projections of $\tilde{s}$ (which is played on $(A_1, \dots, A_{p+1})$) to $(A_1, \dots, A_p)$ and $(A_2, \dots, A_{p+1})$ are plays (resp. interaction sequences). We do this by induction on the number of arenas s is played on:

- if s is played on 1 arena, then there is nothing to prove,
- if s is played on $p+1$ arenas and the duplicated arena is one of the outmost arenas, then one of the projections is s itself (so it is indeed a play) and the other is $\iota(s')$ for some s' played on $p-1$ arenas, so it is a play by induction hypothesis,
- if s is played on $p+1$ arenas and the duplicated arena is none of the outmost arenas, then both projections are in the same case as the second projection in the case above, so they are plays by induction hypothesis.

Finally, to prove that ι_k is a functor, we need to show that if s is a prefix of t, then $\tilde{s}$ is a prefix of $\tilde{t}$, which is direct. □

Proposition 6.3.7. *The category-valued presheaf $\mathbb{P}$ defined by respectively taking $\mathbb{P}_A$, $\mathbb{P}_{A,B}$, $\mathbb{P}_{A,B,C}$ and $\mathbb{P}_{A,B,C,D}$ to be the posets of all justified sequences, plays, interaction sequences and generalised interaction sequences, for all arenas A, B, C, D, with projections and insertions as above, forms a game setting.*

Proof. For $\mathbb{P}$ to be a category-valued presheaf over $\Delta/\mathbb{A}$, one should verify that the simplicial identities hold. They do, but we skip the details.

Copycat plays form a full subcategory and are closed under prefix, hence insertions $\iota_0 : \mathbb{P}_A \to \mathbb{P}_{A,A}$ are discrete fibrations. Projections $\delta_1 : \mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ are discrete fibrations: the restriction of any $u \in \mathbb{P}_{A,B,C}$ along any $p \leq \delta_1(u)$ may be taken to be the shortest prefix of u whose projection is p (longer such prefixes end neither in A nor C). The fact that squares (6.3) are pullbacks is a slight variation on the standard zipping lemma, proved very similarly. Finally, for squares (6.4), it suffices to notice that, if u in $\mathbb{P}_{A,A,B}$ is such that $\delta_2(u) = \iota_0(s)$ for some s in $\mathbb{P}_A$, then $\delta_0(u)$ is mapped to u by ι_0 and to s by δ_1, and similarly in the symmetric case. □

6.3.2 Constraining Strategies

Required: 6.3.1, 2.1.2.
Recommended: $\varnothing$.

In the previous section, we consider a rather rough notion of play. Standardly, further constraints are considered on strategies, such as P-visibility, O-visibility, well-threadedness, and well-bracketing (when games are equipped with an appropriate question-answer discipline). For example, a P-visible strategy is one which only accepts P-visible plays. One then needs to prove that such constraints are *robust*, i.e., are preserved by composition and satisfied by identities. This is done in a very clean and modular way in Harmer's thesis [46, Chapter 3]. In order for our framework to apply to such constrained strategies, we may start from the game setting for unconstrained plays and convert the proof of robustness of a constraint c into the construction of a sub-game setting $\mathbb{P}^c$ that is restricted to constrained plays.

We consider constraints in $\{P\text{-}vis, wb, wt\}$, respectively denoting P-visibility, well-bracketing, and well-threadedness.

Proposition 6.3.8. *Each set $c \subseteq \{P\text{-}vis, wb, wt\}$ of constraints gives rise to a game setting $\mathbb{P}^c$ and an embedding $\mathsf{c}: \mathbb{P}^c \hookrightarrow \mathbb{P}$ of category-valued presheaves.*

Proof. We define each $\mathbb{P}^c_{A,B}$ to be the set of plays defined by our chosen set of constraints c. We then take $\mathbb{P}^c_A$ to be the full subcategory of $\mathbb{P}_A$ consisting of projections of such plays, $\mathbb{P}^c_{A,B,C}$ the full subcategory of $\mathbb{P}_{A,B,C}$ whose projections are in $\mathbb{P}^c_{A,B}$ and $\mathbb{P}^c_{B,C}$, and similarly for $\mathbb{P}^c_{A,B,C,D}$.

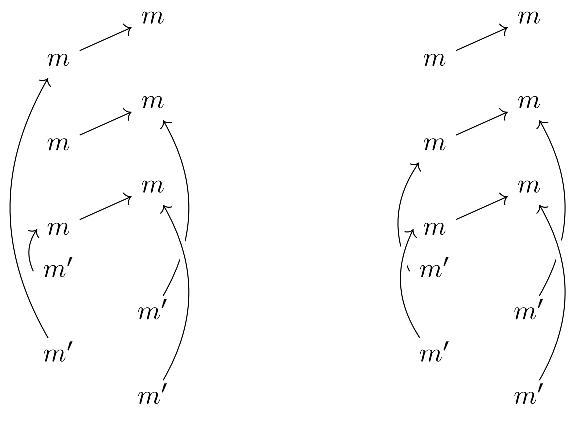
We then check that the morphisms factor through the constrained categories. For projections $\delta_1: \mathbb{P}^c_{A,B,C} \to \mathbb{P}_{A,C}$, this is [46, Proposition 3.4.3] for P-visibility; the implicit proof of [78, Lemma 3.2.4] handles well-bracketing; well-threadedness follows similarly to P-visibility. For insertions, we easily reduce to showing that it is the case for $\iota_0: \mathbb{P}^c_A \to \mathbb{P}_{A,A}$, which is true by definition of $\mathbb{P}^c_A$ (note that $\mathbb{P}^c_A = \mathbb{P}_A$ except when c contains well-bracketing). This yields the desired category-valued presheaf.

It remains to show that it forms a game setting. Projections δ_1 are discrete fibrations because all involved constraints are stable under prefix, so our candidate restriction, defined exactly like in the previous section, is indeed in $\mathbb{P}_{A,B,C}$. We then want to show that constrained plays satisfy zipping. But constraints are merely imposed on the projections of interaction sequences, and thus are clearly stable under zipping. The case of copycat squares is similar. □

Corollary 6.3.9. *For all sets of constraints in $\{P\text{-}vis, wb, wt\}$, arenas and strategies satisfying these constraints form a category.*

Note that we have not claimed that composition in these game settings agrees with composition in our setting for basic HON games. We will show that they do indeed agree later.

Remark. *One could think of treating O-visibility the same way, but this fails for the following reason. The play s on the pair of arenas (A, A) pictured on the left below is O-visible, so we would need its right projection $\delta_0(s)$ to be in $\mathbb{P}^c_A$, or our composition will fail to be unital, because the copycat play $\iota_0(\delta_0(s))$ would not be in $\mathbb{P}_{A,A}$. Unfortunately, that copycat play, which we have drawn on the right below, is not O-visible, so $\delta_0(s)$ cannot be in $\mathbb{P}^c_A$ either...*



At this point, we could consider O-visibility only when plays are already P-visible, which may help (indeed, notice that our example of s is not P-visible).

Proposition 6.3.10. *Each set $c \subseteq \{P\text{-}vis, O\text{-}vis, wb, wt\}$ of constraints containing P-vis gives rise to a game setting $\mathbb{P}^c$ and an embedding $\mathsf{c}{:}\,\mathbb{P}^c \hookrightarrow \mathbb{P}$ of category-valued presheaves.*

Proof. The construction is the same as for Proposition 6.3.8. The implicit proof of [78, Lemma 3.2.4] shows that projections $\delta_1{:}\,\mathbb{P}^c_{A,B,C} \to \mathbb{P}_{A,C}$ factor through $\mathbb{P}^c_{A,C}$. Because copycat strategies are identities for composition, $\iota_0{:}\,\mathbb{P}^c_A \to \mathbb{P}_{A,A}$ factors through $\mathbb{P}^c_{A,A}$. From this, we get that all projections and insertions factor through the constrained categories. □

Beyond the constraints mentioned above, a similar result may be proved for the refined notion of game in McCusker's thesis [78]. Remember that McCusker's games A are just like arenas, except that they come equipped with an abstract *validity* predicate P_A, which is a subset of the set L_A of *legal plays* (P-visible, O-visible, well-bracketed plays), respecting some conditions (see Section 2.1.2 for more details). McCusker then defines $P_{A \multimap B}$ to consist of legal plays in $L_{A \multimap B}$ whose projections to A and B are in P_A and P_B (instead of simply L_A and L_B), respectively. He finally proceeds in a similar way to define interaction sequences and generalised interaction sequences.

In order to organise McCusker's games into a game setting, we should use as a base not mere arenas, but the set $\mathbb{A}^p$ of pairs (A, P_A) of an arena and a predicate on legal plays satisfying the conditions. From the first projection $\mathsf{p}{:}\,\mathbb{A}^p \to \mathbb{A}$, we get the solid part of the cube

$$\begin{tikzcd}
\Delta/\mathbb{A}^p \arrow[rr, "!"] \arrow[dd] \arrow[dr, dashed] \arrow[dr, phantom, "\overset{\beta}{\Longrightarrow}"] & & 1 \arrow[dr, equal] \arrow[dd, "\ulcorner \mathbb{A}^p \urcorner"] & \\
& \Delta/\mathbb{A} \arrow[rr, "!"] \arrow[dd] \arrow[dr, phantom, "\overset{\alpha}{\Longrightarrow}"] & & 1 \arrow[dd, "\ulcorner \mathbb{A} \urcorner"] \\
\Delta \arrow[rr, "\mathsf{i}"'] \arrow[dr, equal] & & \mathsf{Set} \arrow[dr, equal] \arrow[ur, phantom, "\overset{\mathsf{p}}{\Longrightarrow}"] & \\
& \Delta \arrow[rr, "\mathsf{i}"'] & & \mathsf{Set}.
\end{tikzcd}$$

We thus derive a functor $\Delta/\mathsf{p}{:}\,\Delta/\mathbb{A}^p \to \Delta/\mathbb{A}$ as the dashed arrow by universal property of comma, and for any $L \in (\Delta/\mathbb{A}^p)$, we define $\mathbb{P}^p_L$ to be the full subcategory of $\mathbb{P}_{(\Delta/\mathsf{p})(L)}$ spanning plays whose projections satisfy the required predicates. We thus obtain:

Proposition 6.3.11. *The pair $(\mathbb{A}^p, \mathbb{P}^p)$ forms a game setting.*

Proof. The insertions and projections are obviously well-defined. As before, copycat plays form a full subcategory of $\mathbb{P}_{A,A}$, so insertions are discrete fibrations, and projections are discrete fibrations because validity predicates are stable under prefix. It only remains to show that zipping and copycat holds, which again follows from $P_{A \multimap B}$ being only about projections to A and B. □

Corollary 6.3.12. *Games and strategies form a category.*

Beyond this result, we would like to prove that composition and identity in the constrained game settings agree with the original, which is what we will now prove.

Definition 6.3.13. *Given game settings* $(\mathbb{A}, \mathbb{P})$ *and* $(\mathbb{B}, \mathbb{Q})$*, a* morphism *between them consists of a pair of a map* $f : \mathbb{A} \to \mathbb{B}$ *and a natural transformation*

$$\begin{tikzcd}
(\Delta/\mathbb{A})^{op} \arrow[rr, "(\Delta/f)^{op}"] \arrow[dr, "\mathbb{P}"', ""{name=P}] & & (\Delta/\mathbb{B})^{op} \arrow[dl, "\mathbb{Q}", ""'{name=Q}] \\
& \mathsf{Cat}. &
\arrow[from=P, to=Q, Rightarrow, "\alpha", shorten=4mm]
\end{tikzcd} \qquad (6.6)$$

We will denote such a morphism by $\alpha : \mathbb{P} \to \mathbb{Q} \cdot f$.

In such a situation, the functor $\sum_{\alpha_{A,B}} : \widehat{\mathbb{P}_{A,B}} \to \widehat{\mathbb{Q}_{f(A),f(B)}}$ maps strategies in the sense of $(\mathbb{A}, \mathbb{P})$ to strategies in the sense of $(\mathbb{B}, \mathbb{Q})$. Let us prove that under mild hypotheses this functor commutes with composition.

Definition 6.3.14. *We say that a tuple* $(\mathbb{A}, \mathbb{P}, f, \alpha)$ *as above forms a* local morphism *on* $(\mathbb{B}, \mathbb{Q})$ *when*

- *for all* A, B *in* $\mathbb{A}$, $\alpha_{A,B} : \mathbb{P}_{A,B} \to \mathbb{Q}_{f(A),f(B)}$ *is a discrete fibration and*
- *for all* $u \in \mathbb{Q}_{f(A),f(B),f(C)}$, $\alpha^{-1}_{A,B,C}(u) \cong \alpha^{-1}_{A,B}(\delta_2(u)) \times \alpha^{-1}_{B,C}(\delta_0(u))$.

Example 6.3.15. *In any constrained setting, we have a local morphism* $\alpha^c : \mathbb{P}^c \to \mathbb{P}$ *where* $\alpha^{\mathsf{c}}_L : \mathbb{P}^{\mathsf{c}}_L \to \mathbb{P}_L$ *is given by inclusion. Similarly, we have a local morphism* $\alpha^p : \mathbb{P}^p \to \mathbb{P} \cdot \mathsf{p}$ *given again by inclusion (where, we recall,* p *is the functor* $\mathbb{A}^p \to \mathbb{A}$ *that forgets the predicate). Indeed, in both settings, all* α_L*'s are inclusions, hence full embeddings of prefix orderings, and being an interaction sequence is all about the projections being plays, so the second point is verified as well.*

Lemma 6.3.16. *A morphism* (f, α) *as in* (6.6) *whose components are discrete fibrations is a local morphism if and only if* $\partial^*(\alpha_{(A,B),(B,C)})$*, the presheaf induced by the discrete fibration* $\mathbb{P}_{(A,B),(B,C)} \hookrightarrow \mathbb{Q}_{(f(A),f(B)),(f(B),f(C))}$ *is a sheaf for the Grothendieck topology induced by the embedding* $\mathbb{Q}_{f(A),f(B)} + \mathbb{Q}_{f(B),f(C)} \hookrightarrow \mathbb{Q}_{(f(A),f(B)),(f(B),f(C))}$.

Proof. Let us denote by A' the game $f(A)$, etc, and by l and r the embeddings of $\mathbb{Q}_{A',B'}$ and $\mathbb{Q}_{B',C'}$ into $\mathbb{Q}_{(A',B'),(B',C')}$. Let us assume that the morphism is local. To show that $\partial^*(\alpha_{(A,B),(B,C)})$ is a sheaf, let us consider an object c of $Q_{(A',B'),(B',C')}$, a sieve S covering it, and a matching family, and show that there is a unique amalgamation. If c is of the form $l(-)$ or $r(-)$, then S necessarily contains the identity, so there is indeed a unique amalgamation. If c is of the form $m(w)$, then S must contain λ_w and ρ_w. In particular, any matching family x_- thus contains an element $p_{A,B}$ above $l(\delta_2(w))$ and an element $p_{B,C}$ above $r(\delta_0(w))$. These elements are such that $\alpha_{A,B}(p_{A,B}) = \delta_2(w)$ and $\alpha_{B,C}(p_{B,C}) = \delta_0(w)$, so because α is local, we get a unique element $u_{A,B,C}$ such that $\alpha_{A,B,C}(u_{A,B,C}) = w$, which is exactly an element above $m(w)$, which we take as our candidate amalgamation. Proving that it is the unique amalgamation is then routine.

Conversely, if $\partial^*(\alpha_{(A,B),(B,C)})$ is a sheaf, then the unique amalgamation gives the isomorphism between $\alpha^{-1}_{A,B,C}(u)$ and $\alpha^{-1}_{A,B}(\delta_2(u)) \times \alpha^{-1}_{B,C}(\delta_0(u))$. □

In order to explain the discrete fibredness hypothesis, consider the case of a set of constraints c, and imagine that there exists some interaction sequence u in $\mathbb{P}_{A,B,C}$, but not in $\mathbb{P}^c_{A,B,C}$, such that $\delta_2(u)$ and $\delta_0(u)$ are in the respective essential images of $\mathbb{P}^c_{A,B}$ and $\mathbb{P}^c_{B,C}$, say as $\alpha(p_1)$ and $\alpha(p_2)$. Further assuming that σ and τ accept p_1 and p_2, $\mathsf{m}(\sum_\alpha(\sigma), \sum_\alpha(\tau))$ clearly accepts $\delta_1(u)$, while $\sum_\alpha(\mathsf{m}^\alpha(\sigma,\tau))$ does not, because the interaction sequence which could witness it lies outside $\mathbb{P}^c$ (assuming that no interaction sequence from $\mathbb{P}^c$ projects to $\delta_1(u)$).

Proposition 6.3.17. *For any local morphism $\alpha{:}\, \mathbb{P} \hookrightarrow \mathbb{Q}\cdot f$, the following square commutes up to isomorphism.*

$$\begin{tikzcd}
\widehat{\mathbb{P}_{A,B}} \times \widehat{\mathbb{P}_{B,C}} \arrow[r, "\mathsf{m}^{\mathbb{A}}"] \arrow[d, "\Sigma_{\alpha_{A,B}} \times \Sigma_{\alpha_{B,C}}"'] & \widehat{\mathbb{P}_{A,C}} \arrow[d, "\Sigma_{\alpha_{A,C}}"] \\
\widehat{\mathbb{Q}_{f(A),f(B)}} \times \widehat{\mathbb{Q}_{f(B),f(C)}} \arrow[r, "\mathsf{m}^{\mathbb{B}}"'] & \widehat{\mathbb{Q}_{f(A),f(C)}}
\end{tikzcd}$$

Proof. By Proposition 6.2.16, the result reduces to the commutation of

$$\begin{tikzcd}
\mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[r, "\Pi"] \arrow[d, "\Sigma"'] & \mathbb{P}_{(A,B),(B,C)} \arrow[d, "\Sigma"] & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] \arrow[d, "\Sigma"] & \mathbb{P}_{A,C} \arrow[d, "\Sigma"] \\
\mathbb{Q}_{A',B'} + \mathbb{Q}_{B',C'} \arrow[r, "\Pi"'] & \mathbb{Q}_{(A',B'),(B',C')} & \mathbb{Q}_{A',B',C'} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{Q}_{A',C'}
\end{tikzcd}$$

up to isomorphism, where $A' = f(A)$ and so on. The right-hand square commutes up to isomorphism because the underlying square does, the middle one commutes by Lemma 2.2.70, and the left-hand one by Lemmas 6.3.16 and 6.1.2. □

Corollary 6.3.18. *For all set of constraints c, composition in $\mathbb{P}^c$ commutes with embedding into $\mathbb{P}$. Similarly, composition in $\mathbb{P}^p$ commutes with embedding into $\mathbb{P}$.*

There are other kinds of constraints like innocence or single-threadedness, which may not be treated this way. We will deal with innocence in Section 6.4.

6.3.3 AJM Games: a Partial Answer

Required: 6.2.4, 2.1.4.
Recommended: 6.3.1.

Let us now consider the alternative approach to game semantics that are AJM games [6] (see Section 2.1.4 for more details). On the one hand, this approach is more elementary than HON games in that games do not feature justification pointers. On the other hand, games feature a partial equivalence relation between plays, which needs to be dealt with at the level of strategies.

In order to organise such games into a game setting, we have two sensible choices for the notion of morphism between plays: beyond the prefix ordering, we may also incorporate equivalence between plays. Presheaves then amount to so-called *saturated* strategies.

The definitions of $\mathbb{P}_A$ and $\mathbb{P}_{A,B}$ are done exactly as explained in Section 2.1.4. However, we need to slightly change the definition of $\mathbb{P}_{A,B,C}$ for it to fit in our setting. Indeed, if we take $\mathbb{P}_{A,B,C}$ to be the set of all sequences in $M_A + M_B + M_C$ whose projections are plays, then δ_1 may not be a discrete fibration: if s is a prefix of $\delta_1(u)$ and the last moves of u are in B, then there are several prefixes of u whose projection to (A, C) is s, and all these prefixes form an ordered list of interaction sequences, so δ_1 cannot be a discrete fibration. Therefore, we define $\mathbb{P}_{A,B,C}$ similarly to HON games: an interaction sequence is a sequence of moves in $M_A + M_B + M_C$ whose projections are plays and whose last move is in A or C. We define $\mathbb{P}_{A,B,C,D}$ similarly.

Using this definition, we can show that AJM games fit in our framework. However, this has a cost: we will need to be careful when showing that the composition we derive from our framework is the traditional composition in AJM games.

Proposition 6.3.19. *AJM games form a game setting.*

Proof. Insertion functors are obviously well-defined. For projections δ_1, it suffices to notice that $\delta_0(u)$ and $\delta_2(u)$ being plays, their projections are in $\mathbb{P}_A$ and $\mathbb{P}_C$, so $\delta_1(u)$ is a play. Squares (6.3) being pullbacks is once again a slight variation of the standard zipping lemma. The case of squares (6.4) is treated exactly like in HON games. To show that projections $\delta_1 : \mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ are discrete fibrations, we need to canonically restrict any $u \in \mathbb{P}_{A,B,C}$ along any $s \leq \delta_1(u)$, which is done just like in HON games. □

Proposition 6.3.20. *Composition derived from the game setting for AJM games is isomorphic to traditional composition in AJM games.*

Proof. Let us take two strategies $\sigma \in \widehat{\mathbb{P}_{A,B}}$ and $\tau \in \widehat{\mathbb{P}_{B,C}}$ and show that $\sigma;\tau$ accepts a play p if and only if $\overline{\mathsf{m}}(\sigma,\tau)$ does.

If $\overline{\mathsf{m}}(\sigma,\tau)$ accepts p, then there is an interaction sequence u in $\mathbb{P}_{A,B,C}$ such that $\delta_1(u) = p$ and σ accepts $\delta_2(u)$ and τ accepts $\delta_0(u)$. But then, u is also a witness that $\sigma;\tau$ accepts p.

Conversely, we know that $\sigma;\tau$ accepts p if and only if there is an interaction sequence u (but not necessarily in $\mathbb{P}_{A,B,C}$) that projects to p and whose projections are accepted by σ and τ. Then we define $\tilde{u}$ to be the longest prefix of u that ends either in A or C. We have that $\tilde{u}$ is in $\mathbb{P}_{A,B,C}$ because its projections to A and C are equal to those of u, and it is a witness that $\mathsf{m}(\sigma,\tau)$ accepts p. □

For saturated strategies, the idea is to incorporate for all A, B the partial equivalence relations $\approx_A$ and $\approx_B$ into the category of plays.

Proposition 6.3.21. *AJM games form a category-valued presheaf by mapping each list of games to the corresponding set of plays with as morphisms between any two plays u and v:*

- *a singleton when there exists some play w such that $u \approx w \leq v$ (or equivalently there exists w such that $u \leq w \approx v$);*
- *none otherwise.*

However, the obtained category-valued presheaf is not a game setting because projections $\mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ and insertions $\mathbb{P}_A \to \mathbb{P}_{A,A}$ are not discrete fibrations in general. Indeed, the fibres of $\mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ are proper groupoids in general, thus making it a non-discrete Grothendieck fibration. The case of $\mathbb{P}_A \to \mathbb{P}_{A,A}$ is worse: the restriction of a play in $s \in \mathbb{P}_A$ along a morphism $p \to q$ in $\mathbb{P}_{A,A}$ may at best be mapped to some p' isomorphic to p in general, thus making it a *Street fibration*. Our approach may generalise in this direction, but this will involve advanced categorical concepts such as stacks (which are to fibrations as sheaves are to discrete fibrations), so we leave it for future work.

6.3.4 A Non-Example: Blass Games

Required: 6.2.4, 2.1.5.
Recommended: $\varnothing$.

In the previous sections, we have shown that several approaches to game semantics form game settings, with the exception of the saturated AJM setting. It may be instructive to consider Blass's games [14, 15], as they are well-known for their non-associative composition. The reader may refer to Section 2.1.5 for detailed definitions.

For the sake of simplicity, and because it is enough to exhibit a counter-example, let us consider only alternating Blass games, i.e., pairs $A = (T, s)$ of a tree T and a starting player $s \in \{O, P\}$, such that polarity alternates between O and P. For a Blass game $A = (T, s)$, $\mathbb{P}_A$ is the set of vertices of T ordered by position ($x \leq y$ if x is an ancestor of y in T), seen as a category. We define $\mathbb{P}_{A,B}$ as the set of vertices of the game $A \multimap B$, where Proponent has just played. If we imagine plays in as both game trees $A^\perp$ and B drawn side-by-side, then an object of $\mathbb{P}_{A,B}$ is a pair (x, y) of a vertex of $A^\perp$ and one of B that is a child of a pair where it was Proponent's turn to play (i.e., a pair (O, P)). In other words, they are the pairs (x, y) of vertices in $A^\perp$ and B of type (O, O) or (P, P), except maybe the initial pair. We define $\mathbb{P}_{A,B,C}$ similarly.

These definitions are the natural ones given by Blass games. We cannot change the definition of $\mathbb{P}_{A,B}$, because that would change the definition of strategies, and we want our strategies to correspond to standard strategies in Blass games. Similarly, we cannot easily change the definition of $\mathbb{P}_{A,B,C}$, because this definition is at the very base of our definition of composition. However, with these definition, we cannot have both squares (6.3) be pullbacks in general. Indeed, consider the case where the respective polarities of A, B, C and D are O, P, O and P, and A is non-empty. Then, let $\mathbb{P}^l_{A,B,C,D}$ denote the left-hand pullback and $\mathbb{P}^r_{A,B,C,D}$ denote the right-hand one. We will show that both pullbacks cannot be the same category by exhibiting a play in $\mathbb{P}^l_{A,B,C,D}$ which is not in $\mathbb{P}^r_{A,B,C,D}$. First, let us observe that the initial polarities from the respective points of view of $A \multimap B$, $B \multimap C$ and $C \multimap D$ are like so:

$$\begin{tikzcd}
A \arrow[r] & B & B \arrow[r] & C & C \arrow[r] & D \\
A^\perp & B & B^\perp & C & C^\perp & D \\
P & P & O & O & P & P.
\end{tikzcd}$$

Letting a denote any initial move of A, the sequence a is then legal in $\mathbb{P}_{A,B,D}$ (the polarities are PP both in $A \multimap B$ and $A \multimap D$) and the empty sequence is legal in $\mathbb{P}_{B,C,D}$. Thus, a is legal in $\mathbb{P}^l_{A,B,C,D}$ by the left-hand pullback. However, if the two pullbacks were isomorphic, then by the properties of projections

$a \in \mathbb{P}^l_{A,B,C,D}$ would be mapped to $a \in \mathbb{P}^r_{A,B,C,D}$ under the isomorphism. But $\mathbb{P}^r_{A,B,C,D}$ cannot contain a because this play is illegal in $\mathbb{P}_{A,B,C}$ (because the polarity is PO in $A \multimap C$).

6.4 Innocence

6.4.1 Concurrent Innocence

Required: 6.2.3, 2.1.3, 2.2.7.
Recommended: ∅.

In the previous sections, we have constructed a category of games and strategies parameterised over an arbitrary game setting which unifies a number of such categories as instances of the same construction. However, in game models of purely functional languages, the relevant category is the identity-on-objects subcategory of innocent strategies. In this section, we extend game settings with a notion of view, which allows us to construct a subcategory of innocent strategies.

In order to achieve this, we will use the recent recasting of innocence as a sheaf condition [51, 50, 97]. Starting from HON games, the first step is to refine the posets $\mathbb{P}_A$, $\mathbb{P}_{A,B}$, and $\mathbb{P}_{A,B,C,D}$ into proper categories (with exactly the same objects), say $\mathbb{P}^+_A$, $\mathbb{P}^+_{A,B}$, and $\mathbb{P}^+_{A,B,C,D}$, with the crucial feature that for any play $p \in \mathbb{P}^+_{A,B}$ and index i of a move in p, there is a morphism $\lceil p \rceil_i \to p$. This of course does not hold with the prefix ordering, as the view is rarely a prefix. This idea was introduced in [80] in a slightly different setting.

Passing from $\mathbb{P}$ to $\mathbb{P}^+$ raises the issue of how to extend the abstract framework. Should it now contain two category-valued presheaves? Or should we simply forget about prefix-based strategies and accept $\mathbb{P}^+$ as the new basic set up? We do not make any definitive choice here, but for simplicity and modularity reasons, we choose to first work with $\mathbb{P}^+$ only, and introduce $\mathbb{P}$ in a second stage.

Indeed, perhaps surprisingly, the approach using simply $\mathbb{P}^+$ works for Tsukada and Ong's model. If we take $\mathbb{P}^+_L$ to be the categories with the same objects as $\mathbb{P}_L$ for HON games with the P-visibility constraint, but whose morphisms are given by block-preserving maps (see Section 2.1.3 for more details), we have:

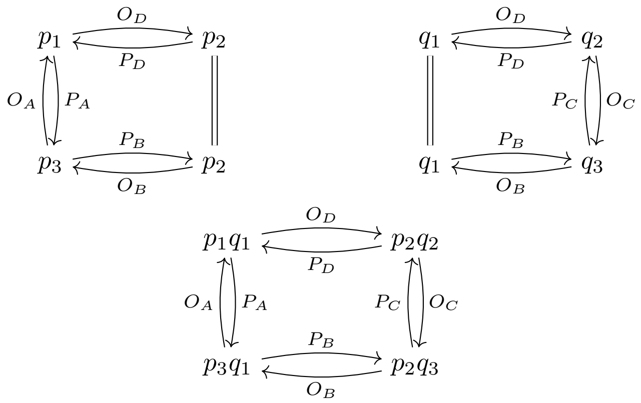
Proposition 6.4.1. *Tsukada and Ong's $\mathbb{P}^+$ forms a game setting.*

Proof. This mainly follows from Lemmas 39, 46 and 47 of [96] or is obvious. The only non-obvious point that is not shown by these lemmas is that the zipping squares are pullbacks (because Lemma 47 is only about objects). Noticing that the zipping lemma is all about choosing a good pushout will save us some trouble. Indeed, if $u = (n, f, \varphi)$ is in $\mathbb{P}_{A,B,D}$ and $v = (m, g, \psi)$ is in $\mathbb{P}_{B,C,D}$ and their projections to (B, D) are equal, they may be represented as the back faces of the cube below, and the zipping w of u and v may be seen as a good choice of pushout, compatible with pointers, and such that the maps $n \to q$ and $m \to q$ are increasing, as in

$$\begin{tikzcd}
p \arrow[rr] \arrow[dr] \arrow[dd] & & m \arrow[dr] \arrow[dd, "g"] & \\
& n \arrow[rr] \arrow[dd, "f"] & & q \arrow[dd, "h"] \arrow[ul, phantom, "\ulcorner", very near start] \\
M_{B,D} \arrow[rr] \arrow[dr] & & M_{B,C,D} \arrow[dr] & \\
& M_{A,B,D} \arrow[rr] & & M_{A,B,C,D}, \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

where p is the length of $\delta_0(u)$. Such a square is a pushout if and only if (i) n and m are jointly surjective and (ii) i in q is in the image of both n and m if and only if it is in the image of p (because all the relevant maps are monic). This pushout (or rather, the maps $n \to q$ and $m \to q$) is standardly given by the zipping lemma (although it is usually not explained from this point of view). We are going to build maps $n \to q$ and $m \to q$ that make the square above a pushout as is done in the zipping lemma.

One builds $n \to q$ and $m \to q$ by induction on (n, m), by looking at the switching automata for $\mathbb{P}_{A,B,D}$ and $\mathbb{P}_{B,C,D}$ and combining them into the switching automaton for $\mathbb{P}_{A,B,C,D}$. The different automata are as drawn below.



The automata respectively start in state p_1, q_1, and p_1q_1, and to know the antecedents of $k+1$ in q, we just keep track of the biggest i in n and j in m that already have an image as well as the state pq our automaton is in, and, based on that information, choose whether $i+1$ or $j+1$ (or both) are mapped to $k+1$. An induction invariant is that i and j are such that $\delta_0(u_{|i}) = \delta_1(v_{|j})$. For example, if the automaton is in the state p_1q_1, then the first automaton is in state p_1, so the next move in u is either an Opponent move in D or a Proponent move in A, and the second automaton is in state q_1, so the next move in v is either an Opponent move in D or a Proponent move in B. But we know that both projections to (B, D) are equal up to i and j, so if the next move of u is in D, then so is the next move of v, and that is what the zipping chooses (both $i+1$ and $j+1$ are mapped to $k+1$). Otherwise, the next move in u is a Proponent move in A, and that is what the zipping chooses (only $i+1$ is mapped to $k+1$). Pointers in w are inherited from those in u and v (which is well-defined because these pointers agree on (B, D)). To prove that the sequence thus defined is the standard zipping of u and v, we need to show that it is a justified sequence (which is mostly obvious because the pointers are inherited from u and v) and that its projections to (A, B, D) and (B, C, D) are

u and v, which is by adhesivity (Lemma 4.2.5). Indeed, both horizontal faces are pushouts along monos, and the back faces are pullbacks because they are projections, so the front faces are also pullbacks.

Now, we want to show that the same idea can be used to show that

$$\begin{tikzcd}
\mathbb{P}_{A,B,C,D} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{P}_{A,B,D} \arrow[d] \\
\mathbb{P}_{B,C,D} \arrow[r] & \mathbb{P}_{B,D}
\end{tikzcd}$$

is a pullback on morphisms as well. The idea is that, if $u = (n, f, \varphi)$ and $u' = (n', f', \varphi')$, a morphism $u \to u'$ is simply a map $n \to n'$ satisfying some additional conditions. In particular, if we have two morphisms $\alpha: u \to u'$ and $\beta: v \to v'$ whose projections to (B, D) are equal, then we have a diagram as the solid part of

$$\begin{tikzcd}
p \arrow[rr] \arrow[dr] \arrow[dd] & & m \arrow[dr, "\mathrm{inr}"] \arrow[dd, "\beta"] & \\
& n \arrow[rr, "\mathrm{inl}"] \arrow[dd, "\alpha"] & & q \arrow[dd, dashed, "\gamma"'] \arrow[dddd, bend left, "h"] \\
p' \arrow[rr] \arrow[dr] \arrow[dd] & & m' \arrow[dr, "\mathrm{inr}'"] \arrow[dd, "g'"] & \\
& n' \arrow[rr, "\mathrm{inl}'"] \arrow[dd, "f'"] & & q' \arrow[dd, "h'"'] \\
M_{B,D} \arrow[rr] \arrow[dr] & & M_{B,C,D} \arrow[dr] & \\
& M_{A,B,D} \arrow[rr] & & M_{A,B,C,D},
\end{tikzcd}$$

where $\gamma: q \to q'$, obtained by universal property of pushout, is our candidate morphism from w to w'. First, let us notice that this is the only possible choice of γ because inl and inr are jointly surjective. Another point that can directly be checked is that the projections of γ are indeed α and β (again by adhesivity, as above). Therefore, the only thing left to show is that γ is a morphism from w to w', which means that it is injective, compatible with h and h', pointers, and maps basic blocks to basic blocks. The first point is by extensivity (Lemma 4.2.6). The second is by uniqueness in the universal property of q. For compatibility with pointers, if we call χ the pointer function w inherits from α and β (and similarly χ' for w'), inl and inr being jointly surjective, we may compute, for moves i in $A + P_B + D$ (i.e., for moves i such that i and $\varphi(i)$ are in $A + B + D$)

$$\begin{aligned}
\gamma(\chi(\mathrm{inl}(i))) &= \gamma(\mathrm{inl}(\varphi(i))) && \text{by definition of } \chi \\
&= \mathrm{inl}'(\alpha(\varphi(i))) \\
&= \mathrm{inl}'(\varphi'(\alpha(i))) \\
&= \chi'(\mathrm{inl}'(\alpha(i))) && \text{by definition of } \chi' \\
&= \chi'(\gamma(\mathrm{inl}(i))),
\end{aligned}$$

where the penultimate equality comes from the fact that the definition of χ' only depends on the state of the automaton and the "types" of u'_{i+1} and v'_{j+1}, and the existence of α and β ensure that we are in the same state of the automaton when building $k + 1$ (in w) and $\gamma(k + 1)$ (in w'). A similar computation shows that γ also preserves pointers for moves of the form $\mathrm{inr}(j)$ for moves in $B + C + D$.

Finally, we must show that γ preserves basic blocks, i.e., that if $h(k)$ is in $P_A + M_B + M_C + O_D$, then $\gamma(k + 1) = \gamma(k) + 1$. Each k corresponds to an arrow in the switching automaton, which is the arrow taken while building h. In the switching automaton on (A, B, C, D), the horizontal arrows correspond to steps in the construction of h where k has an antecedent in n and one in m (and they correspond to certain pairs of arrows in both switching automata), the left-hand side ones correspond to steps where k only has an antecedent in n (and to certain arrows of the switching automaton on (A, B, D)) and the other ones to steps where k only has an antecedent in m. Consider any k such that $h(k)$ is in $P_A + M_B + M_C + O_D$. It corresponds to one of the arrows, but cannot be the O_A or P_D arrows. If it corresponds to an arrow that points to one of the right-hand states, we know that k has an antecedent j in m, and we have $\gamma(k + 1) = \gamma(\mathrm{inr}(j) + 1) = \gamma(\mathrm{inr}(j + 1))$ because all arrows from some right-hand state come from the switching automaton on (B, C, D), so we know the next move also has an antecedent in m. But β preserves basic blocks and $g(j) = h(\mathrm{inr}(j)) = h(k)$ is in $P_B + C + O_D$ (because we only took the arrows that point to one of the right-hand states), so $\beta(j + 1) = \beta(j) + 1$. Now, because k corresponds to an arrow that points to a right-hand state, we know that the next arrow also comes from the switching automaton for (B, C, D), so $\mathrm{inr}'(\beta(j) + 1) = \mathrm{inr}'(\beta(j)) + 1$. This finally gives the desired result because $\mathrm{inr}'(\beta(j)) = \gamma(\mathrm{inr}(j))$. The proof is similar when we take one of the arrows that point to one of the left-hand states. □

Returning to the abstract setting, the new data thus merely consists of a full subcategory $\mathsf{i}_{A,B} : \mathbb{V}_{A,B} \hookrightarrow \mathbb{P}_{A,B}$, for all A, B, whose objects are called *views*.

Definition 6.4.2. *The category of* innocent *strategies is the essential image of* $\prod_{\mathsf{i}_{A,B}} : \widehat{\mathbb{V}_{A,B}} \to \widehat{\mathbb{P}_{A,B}}$ *(or equivalently the category of sheaves for the Grothendieck topology induced by* $\mathsf{i}_{A,B}$*, by Lemma 2.2.84). The domain* $\widehat{\mathbb{V}_{A,B}}$ *is the category of* behaviours*.*

We now would like to establish that in any game setting equipped with such full embeddings, innocent strategies form a subcategory. However, our proof relies on two additional properties. The first one, already observed in [97, Lemma 32], states that one can reconstruct uniquely any interaction sequence from its projection to $\mathbb{P}_{A,C}$, say u, together with a compatible family, indexed by all views v of u, of interaction sequences projecting to v. The second property essentially says that any morphism $v \to \delta_2(u)$ from a view $v \in \mathbb{V}_{A,B}$ to the projection of some interaction sequence $u \in \mathbb{P}^{+}_{A,B,C}$ factors canonically through $\delta_2(w)$ of some interaction sequence w whose projection $\delta_1(w)$ is a view (and similarly for δ_0).

Let us introduce both properties in more detail.

The first property essentially says that interaction is local. The projection $\mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$, as a discrete fibration, induces a presheaf $\partial^*(\delta_1)$ on $\mathbb{P}_{A,C}$ which we will require to be in the essential image of $\prod_{\mathsf{i}_{A,B}} : \widehat{\mathbb{V}_{A,C}} \to \widehat{\mathbb{P}_{A,C}}$. By Lemma 2.2.84, this is equivalent to requiring that $\partial^*(\delta_1)$ be a sheaf for the Grothendieck topology induced by the embedding $\mathsf{i}_{A,C}$. Similarly, we require the presheaf induced by $\iota_0 : \mathbb{P}_A \to \mathbb{P}_{A,A}$ to be a sheaf for the Grothendieck topology induced by the embedding $\mathbb{V}_{A,A} \to \mathbb{P}_{A,A}$. Let us record this as:

Definition 6.4.3. *A game setting* $(\mathbb{A}, \mathbb{P})$ *equipped with full embeddings* $\mathsf{i}_{A,B}$ *from* $\mathbb{V}_{A,B}$ *to* $\mathbb{P}_{A,B}$ *is* local *if and only if* $\partial^*(\delta_1)$ *and* $\partial^*(\iota_0)$ *are sheaves.*

Proposition 6.4.4. *Tsukada and Ong's* $\mathbb{P}^+$ *is local.*

Proof. For δ_1, the result is precisely [97, Lemma 32]. For ι_0, just observe that a play is copycat if and only if all its views are. □

So locality is the first property we need to require of our game settings with views. The second property has to do with projections, e.g., $\delta_2 : \mathbb{P}_{A,B,C} \to \mathbb{P}_{A,B}$. It essentially says that any morphism $v \to \delta_2(u)$ with $v \in \mathbb{V}_{A,B}$ and $u \in \mathbb{P}_{A,B,C}$ factors "canonically" through some $\delta_2(w)$ with $w \in \mathbb{V}_{A,B,C}$, where $\mathbb{V}_{A,B,C}$ is the pullback

$$\begin{tikzcd}
\mathbb{V}_{A,B,C} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathbb{P}_{A,B,C} \arrow[d, "\delta_1"] \\
\mathbb{V}_{A,C} \arrow[r, "\mathsf{i}_{A,C}"'] & \mathbb{P}_{A,C}.
\end{tikzcd}$$

In order to define such canonicity, we appeal to the theory of analytic functors [59, 98, 99].

Definition 6.4.5 (Weber [98, 99])**.** *A functor* $T : \mathbb{C} \to \mathbb{D}$ *admits* generic factorisations *relative to an object* $d \in \mathbb{D}$ *if and only if any* $f : d \to Tc$ *admits a factorisation as below left*

$$\begin{tikzcd}
d \arrow[d, "g"'] \arrow[dr, "f"] & \\
Ta \arrow[r, "Th"'] & Tc
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
d \arrow[r, "g'"] \arrow[d, "g"'] & Tb \arrow[d, "Th'"] \\
Ta \arrow[r, "Th"'] \arrow[ur, dashed, "Tk"] & Tc
\end{tikzcd}$$

such that for all commuting squares as the exterior above right, there exists a lifting k *as shown such that* $g' = Tk \circ g$ *and* $h = h'k$*. The middle object* a *is called the* arity *of* f *– all generic factorisations share the same* a *up to isomorphism.*

For all subcategories $\mathbb{B} \hookrightarrow \mathbb{C}$ *and* $\mathbb{E} \hookrightarrow \mathbb{D}$*, a functor* $\mathbb{C} \to \mathbb{D}$ *admitting generic factorisations relative to all objects of* $\mathbb{E}$ *with arities in* $\mathbb{B}$ *is called* $(\mathbb{B}, \mathbb{E})$-analytic *[39].*

Definition 6.4.6. *A game setting* $(\mathbb{A}, \mathbb{P})$ *equipped with full embeddings* $\mathsf{i}_{A,B}$ *from* $\mathbb{V}_{A,B}$ *to* $\mathbb{P}_{A,B}$ *is* view-analytic *when* δ_2 *is* $(\mathbb{V}_{A,B,C}, \mathbb{V}_{A,B})$*-analytic and* δ_0 *is* $(\mathbb{V}_{A,B,C}, \mathbb{V}_{B,C})$*-analytic.*

Proposition 6.4.7. *Tsukada and Ong's game setting [97] is view-analytic.*

The following proof is due to Takeshi Tsukada (private communcation):

Proof. Let us take a play $u = (n, f, \varphi)$ in $\mathbb{P}_{A,B,C}$, a view $v = (p, g, \psi)$ in $\mathbb{V}_{A,B}$, and a morphism $\alpha : v \to \delta_2(u)$. We want to show that there exists w in $\mathbb{P}_{A,B,C}$ whose projection $\delta_1(w)$ is in $\mathbb{V}_{A,C}$ and morphisms $\beta : v \to \delta_2(w)$ and $\gamma : w \to u$ such that $\alpha = \delta_2(\gamma) \circ \beta$. We further want that, for any other such w', β', and γ', there exists a morphism $\zeta : w \to w'$ such that $\delta_2(\zeta) \circ \beta = \beta'$ and $\gamma' \circ \zeta = \gamma$. (We only treat the case of δ_2, proving the same properties for δ_0 follows exactly the same pattern.)

For all i in n, let us define the sequence $w^i_{A,B,C}$ of indices of u by induction as follows:

$$\begin{aligned} w^0_{A,B,C} &= \varepsilon, \\ w^i_{A,B,C} &= \begin{cases} w^{i-1}_{A,B,C} \cdot i & \text{if } f(i) \text{ is in } O_A,\ M_B, \text{ or } P_C \\ w^{\varphi(i)}_{A,B,C} \cdot i & \text{if } f(i) \text{ is in } P_A \text{ or } O_C. \end{cases} \end{aligned}$$

Our candidate w will more or less be $w^{\alpha(p)}_{A,B,C}$ (this is slightly wrong because it is not always exactly an interaction sequence). To prove that v embed into w, we also define the families $v^i_{A,B}$ and $v^i_{B,C}$ of sequences, which are only defined if $f(i)$ is in $M_A + M_B$ or $M_B + M_C$ respectively:

$$\begin{aligned} v^i_{A,B} &= \begin{cases} i & \text{if } f(i) \text{ is in } O_B \text{ and } f(\varphi(i)) \text{ is in } M_C \\ v^{i-1}_{A,B} \cdot i & \text{if } f(i) \text{ is in } O_A \text{ or } P_B \\ v^{\varphi(i)}_{A,B} \cdot i & \text{if } f(i) \text{ is in } P_A \text{ or } O_B \text{ (and } f(\varphi(i)) \text{ is not in } M_C), \end{cases} \\ v^0_{B,C} &= \varepsilon, \\ v^i_{B,C} &= \begin{cases} v^{i-1}_{B,C} \cdot i & \text{if } f(i) \text{ is in } O_B \text{ or } P_C \\ v^{\varphi(i)}_{B,C} \cdot i & \text{if } f(i) \text{ is in } P_B \text{ or } O_C. \end{cases} \end{aligned}$$

We now want to prove the following properties as preliminary work towards the result:

(i) $\alpha(v) = v^{\alpha(p)}_{A,B}$ (where α of a sequence is its application to each move in the sequence),

(ii) for all i in n, $v^i_{A,B}$ and $v^i_{B,C}$ are previews and included in $w^i_{A,B,C}$ (when they are defined),

(iii) for all i in n, $w^i_{A,B,C}$ is a pre-interaction sequence (a sequence whose left and right projections are preplays),

(iv) for all i in n, $\delta_1(w^i_{A,B,C})$ is a preview.

We prove *(i)* by induction on the length of v: if v is only one move, then the result is obvious, otherwise $v = v' \cdot m$ with $v' = (p', g', \psi')$ a view. If $p' + 1$ is even, then

$$\begin{aligned} v^{\alpha(p'+1)}_{A,B} &= v^{\alpha(p'+1)-1}_{A,B} \cdot \alpha(p'+1) \\ &= v^{\alpha(p')}_{A,B} \cdot \alpha(p'+1) && \text{by preservation of } OP\text{-blocks} \\ &= \alpha(v) && \text{by induction hypothesis.} \end{aligned}$$

When $p' + 1$ is odd, we have

$$\begin{aligned}
v^{\alpha(p'+1)}_{A,B} &= v^{\varphi(\alpha(p'+1))}_{A,B} \cdot \alpha(p'+1) \\
&= v^{\alpha(\psi(p'+1))}_{A,B} \cdot \alpha(p'+1) \\
&= v^{\alpha(p')}_{A,B} \cdot \alpha(p'+1) && \text{because all Opponent moves point} \\
& && \text{to their predecessors in views} \\
&= \alpha(v) && \text{by induction hypothesis.}
\end{aligned}$$

For *(ii)*, proving that $v^i_{A,B}$ and $v^i_{B,C}$ are views is a simple induction on i: it suffices to show that all Opponent moves point to their predecessors, which is direct by definition. To prove that they are included in $w^i_{A,B,C}$, we proceeed by induction on i. If i is 0, then the result is obvious. Otherwise, if $v^i_{A,B}$ is defined (otherwise there is nothing to prove), then the result is obvious when $f(i)$ is in O_A, P_A, or P_B (because $v^i_{A,B}$ and $w^i_{A,B,C}$ follow the "same" recursive definition) and when $f(i)$ is in O_B and $f(\varphi(i))$ in M_C (because this is the base case for $v^i_{A,B}$). When $f(i)$ is in O_B and $f(\varphi(i))$ is not in M_C, then $v^i_{A,B} = v^{\varphi(i)}_{A,B} \cdot i \subseteq w^{\varphi(i)}_{A,B,C} \cdot i$ by induction hypothesis, but $\delta_0(u)$ is a play and thus i, being a Proponent move (in $\delta_0(u)$) must point into $\lceil \delta_0(u) \rceil_{i-1} = v^{i-1}_{B,C}$ (because $v^{i-1}_{B,C}$ is a view and its last move is $i-1$). But then $\varphi(i) \in v^{i-1}_{B,C} \subseteq w^{i-1}_{A,B,C}$ by induction hypothesis, so necessarily $w^{\varphi(i)}_{A,B,C} \subseteq w^i_{A,B,C}$, and finally $v^i_{A,B} \subseteq w^i_{A,B,C}$ (the case of $v^i_{B,C}$ is similar).

For *(iii)*, we also proceed by induction on i. The projections are obviously alternating and justified, and the proof of P-visibility follows directly from *(ii)*.

Finally, *(iv)* is obvious by definition of $w^i_{A,B,C}$ since the beginning of each basic block points to the end of the preceding one.

We can now put all this together to prove the result. Our candidate w is $w^{\alpha(p)}_{A,B,C}$, augmented with the moves of the basic block containing $\alpha(p)$ from $\alpha(p)+1$ to the end of the block. By *(iii)*, w is an interaction sequence (it is a pre-interaction sequence, all Proponent moves point into their views in both projections (same trick as in the proof of *(ii)*) and it ends in either A or C because we augmented it with the last moves of the block). Its projection to (A,C) is a view by *(iv)*, because we only possibly added the last move of the last basic block to $w^{\alpha(p)}_{A,B,C}$, and that last move cannot change the view (it is a Proponent move). We choose $\beta\colon v \to \delta_2(w)$ to be the obvious morphism induced by α, which exists by *(i)* and *(ii)*, and we choose $\gamma\colon w \to u$ to be given by inclusion. We obviously have that $\alpha = \delta_2(\gamma) \circ \beta$. The existence of the ζ morphism for any other such decomposition comes from the fact that w is actually the smallest interaction sequence that is a view when projected to (A,C), is contained in u, and contains v (this is direct from the definition of $w^i_{A,B,C}$). □

We may now state our main result about innocence:

Definition 6.4.8. *An* innocent game setting *is a game setting* $(\mathbb{A},\mathbb{P})$ *equipped with full embeddings* $\mathsf{i}_{A,B}\colon \mathbb{V}_{A,B} \hookrightarrow \mathbb{P}_{A,B}$*, which is both local and view-analytic.*

Theorem 6.4.9. *In any innocent game setting, innocent strategies form a subcategory.*

We prove this theorem in two lemmas. The first states stability of innocence under composition; the second says that copycat strategies are innocent.

Lemma 6.4.10. *In any innocent game setting, the composite of two innocent strategies is again innocent.*

Proof. Because a strategy X, say on (A, B), is innocent if and only if it is isomorphic to $\prod_{\mathsf{i}_{A,B}}(\Delta_{\mathsf{i}_{A,B}}(X))$, it suffices to show that starting from any pair of behaviours $[B_1, B_2] \in \widehat{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}}$, if we extend them to innocent strategies, compose, and eventually apply innocentisation (i.e., $\prod_{\mathsf{i}_{A,C}} \circ \Delta_{\mathsf{i}_{A,C}}$), then the last step is redundant. In other words, we need to show that the perimeter of

$$\begin{tikzcd}
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[r, "\Pi"] \arrow[d, equal] & \mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[r, "\Pi"] & \mathbb{P}_{(A,B),(B,C)} & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] & \mathbb{P}_{A,C} \\
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[rr, "\Pi"] \arrow[d, equal] & & \mathbb{V}_{(A,B),(B,C)} \arrow[u, "\Delta"] \arrow[d, "\Pi"'] & \mathbb{V}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] \arrow[u, "\Delta"] \arrow[d, "\Pi"'] & \mathbb{V}_{A,C} \arrow[u, "\Delta"'] \arrow[d, "\Pi"] \\
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{(A,B),(B,C)} & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{P}_{A,C}
\end{tikzcd}$$

commutes up to isomorphism. We proceed by showing that innocentisation is redundant at every intermediate step, but this requires us to define intermediate categories of views adequately. In particular, to define $\mathbb{V}_{(A,B),(B,C)}$, consider first the lax colimit $\mathbb{C}$ of $\mathbb{P}_{A,B} \leftarrow \mathbb{V}_{A,B,C} \rightarrow \mathbb{P}_{B,C}$, and then restrict it to its full subcategory spanning objects from $\mathbb{V}_{A,B,C}$, $\mathbb{V}_{A,B}$, and $\mathbb{V}_{B,C}$.

Returning to our claim, the top-left square commutes by Lemma 2.2.68, the top square because the underlying diagram commutes, the top-right square by Lemma 2.2.70, the bottom-left square because the underlying diagram commutes, and the bottom-right one by locality and Lemma 6.1.2.

Let us finally show that the bottom square is exact, using the zigzag criterion and view-analyticity. Let us take v in $\mathbb{V}_{(A,B),(B,C)}$, $u_{A,B,C}$ in $\mathbb{P}_{A,B,C}$, and a morphism $f: v \to m(u_{A,B,C})$ in $\mathbb{P}_{(A,B),(B,C)}$. If v has the form $m(v_{A,B,C})$, then the zigzag criterion holds directly by taking $v_{A,B,C}$. Otherwise, if v is of the form $l(v_{A,B})$, then f is of the form $l(v_{A,B}) \xrightarrow{l(f_0)} l(\delta_2(u_{A,B,C})) \xrightarrow{\lambda_{u_{A,B,C}}} m(u_{A,B,C})$, which factors through some $l(\delta_2(v_{A,B,C}))$ by view-analyticity, and the zigzag criterion holds. The proof is similar if v is of the form $r(v_{B,C})$. □

We finally prove innocence of identities:

Lemma 6.4.11. *Copycat strategies are innocent.*

Proof. We proceed as for preservation of innocence by composition: by showing that copycat is the same as copycat followed by innocentisation. This yields the diagram

$$\begin{tikzcd}
\varnothing \arrow[r, "\Pi"] \arrow[d, equal] & \mathbb{P}_A \arrow[r, "\Sigma"] & \mathbb{P}_{A,A} \\
\varnothing \arrow[r, "\Pi"] \arrow[d, equal] & \mathbb{V}_A \arrow[u, "\Delta"] \arrow[d, "\Pi"'] \arrow[r, "\Sigma"] \arrow[ur, phantom, "\urcorner", very near start] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{V}_{A,A} \arrow[u, "\Delta"'] \arrow[d, "\Pi"] \\
\varnothing \arrow[r, "\Pi"'] & \mathbb{P}_A \arrow[r, "\Sigma"'] & \mathbb{P}_{A,A},
\end{tikzcd}$$

where $\mathbb{V}_A$ is defined as the marked pullback. The bottom-left square commutes because underlying functors do, the top-left one commutes by Lemma 2.2.68, the bottom-right one commutes by locality and Lemma 6.1.2, and the top-right one by Lemma 2.2.70. □

6.4.2 Prefix-Based Innocence

Required: 6.4.1.
Recommended: ∅.

In the previous section, we have shown that innocent strategies behave well in any innocent game setting. However, our only concrete example of an innocent game setting for now is Tsukada and Ong's $\mathbb{P}^+$. There is in fact a further example, given by enriching arenas with bracketing information and restricting $\mathbb{P}^+_{A,B}$ to well-bracketed plays [97, Section VII]. This shows that innocence is stable under cg-composition. But we are also interested in pb-composition of innocent strategies, since most of the examples are of this form. As mentioned before, innocence is not stable under pb-composition in general [46, Section 3.7.2], unless one restricts to deterministic strategies. In an attempt to better understand this phenomenon, we first move in this section from cg-composition to pg-composition, and prove that innocence remains stable there. In the next section, we will explain why this does not carry over to pb-composition, although, as is well-known, it does on deterministic strategies.

We proceed by first defining innocent pg-strategies in an extended framework and then showing that our definition extends the standard one. We then show that pg-innocent strategies include copycats and are closed under composition.

Definition 6.4.12. *Consider game settings $(\mathbb{A}, \mathbb{P}^+)$ and $(\mathbb{A}, \mathbb{P})$ with the same set of arenas and $\mathbb{V}$ making $\mathbb{P}^+$ innocent, further equipped with a componentwise identity-on-objects natural embedding $\mathsf{k} \colon \mathbb{P} \hookrightarrow \mathbb{P}^+$ such that each $\mathsf{i}_{A,B} \colon \mathbb{V}_{A,B} \hookrightarrow \mathbb{P}^+_{A,B}$ factors through $\mathsf{k}_{A,B}$. Let a presheaf on $\mathbb{P}_{A,B}$ be* innocent via $\mathbb{P}^+$, *or* $\mathbb{P}^+$-innocent, *if and only if it is in the essential image of $\widehat{\mathbb{V}_{A,B}} \xrightarrow{\Pi_{\mathsf{i}_{A,B}}} \widehat{\mathbb{P}^+_{A,B}} \xrightarrow{\Delta_{\mathsf{k}_{A,B}}} \widehat{\mathbb{P}_{A,B}}$. Similarly, let a presheaf on $\mathbb{P}_{A,A}$ be* $\mathbb{P}^+$-copycat *if and only if it is in the essential image of $1 \cong \widehat{\varnothing} \xrightarrow{\Pi_!} \widehat{\mathbb{P}^+_A} \xrightarrow{\Sigma_{\iota_0}} \widehat{\mathbb{P}^+_{A,A}} \xrightarrow{\Delta_{\mathsf{k}_{A,A}}} \widehat{\mathbb{P}_{A,A}}$.*

In such a setting, one could consider studying $(\mathbb{V}, \mathbb{P})$ directly. However, it will rarely form an innocent game setting (essentially only when k is an isomorphism).

Example 6.4.13. *Let us take the classical HON setting with P-visible plays, which only differs from Tsukada and Ong's setting by its morphisms. A matching family for a play p in $\mathbb{P}_{A,C}$ is then the data, for all views v that are a prefix of p, of an interaction sequence u_v that projects to v, and such that u_v is a prefix of $u_{v'}$ when v is a prefix of v'. Matching families for p are thus isomorphic to interaction sequences that project to the longest prefix of p that is a view. If $\partial^*(\delta_1)$ were a sheaf, it would mean that there is a single way to amalgamate matching families. In other words, for all plays p and interaction sequence u_v that projects to the longest prefix v of p that is a view, there should be a unique u that projects to p and that admits u_v as a prefix, which is clearly false in general.*

Let us now state the transfer result.

Proposition 6.4.14. *In the setting of Definition 6.4.12, if all naturality squares*

$$\begin{tikzcd}
\mathbb{P}_{A,B,C} \arrow[r] \arrow[d] & \mathbb{P}_{A,C} \arrow[d] & \text{and} & \mathbb{P}_{A} \arrow[r] \arrow[d] & \mathbb{P}_{A,A} \arrow[d] \\
\mathbb{P}^{+}_{A,B,C} \arrow[r] & \mathbb{P}^{+}_{A,C} & & \mathbb{P}^{+}_{A} \arrow[r] & \mathbb{P}^{+}_{A,A}
\end{tikzcd}$$

are pullbacks, then $\mathbb{P}^{+}$-innocent strategies are closed under composition and comprise $\mathbb{P}^{+}$-copycat strategies, which are exactly copycat strategies.

Proof. We proceed as in the previous section: we first need to show that the exterior of the following diagram commutes up to isomorphism

$$\begin{tikzcd}
& \mathbb{P}_{A,B}+\mathbb{P}_{B,C} \arrow[r, "\Pi"] \arrow[d, "\Delta"'] & \mathbb{P}_{(A,B),(B,C)} \arrow[d, "\Delta"'] & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] \arrow[d, "\Delta"'] & \mathbb{P}_{A,C} \arrow[d, "\Delta"'] \\
\mathbb{V}_{A,B}+\mathbb{V}_{B,C} \arrow[r, "\Pi"] \arrow[dd, equal] & \mathbb{P}^{+}_{A,B}+\mathbb{P}^{+}_{B,C} \arrow[r, "\Pi"] & \mathbb{P}^{+}_{(A,B),(B,C)} & \mathbb{P}^{+}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] & \mathbb{P}^{+}_{A,C} \\
& & \text{(Theorem 6.4.9)} & & \mathbb{V}_{A,C} \arrow[u, "\Delta"] \arrow[uu, bend right, "\Delta"'] \arrow[d, "\Pi"] \\
\mathbb{V}_{A,B}+\mathbb{V}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}^{+}_{A,B}+\mathbb{P}^{+}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}^{+}_{(A,B),(B,C)} & \mathbb{P}^{+}_{A,B,C} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{P}^{+}_{A,C} \\
& \mathbb{P}_{A,B}+\mathbb{P}_{B,C} \arrow[r, "\Pi"'] \arrow[u, "\Delta"] & \mathbb{P}_{(A,B),(B,C)} \arrow[u, "\Delta"] & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] \arrow[u, "\Delta"] & \mathbb{P}_{A,C}, \arrow[u, "\Delta"']
\end{tikzcd}$$

where the diagram should be read starting from $\mathbb{V}_{A,B}+\mathbb{V}_{B,C}$. Indeed, the top path corresponds to composition of two $\mathbb{P}^{+}$-innocent strategies followed by innocentisation, while the bottom one simply corresponds to composition. Because the triangle and top square commute (because the underlying diagrams do) and the bottom part of the diagram is exactly the same as the top one, by Theorem 6.4.9, this reduces to exactness of the top-left and top-right squares. The former follows by construction of $\mathbb{P}^{+}_{(A,B),(B,C)}$, using the zigzag criterion. The latter follows by discrete fibredness of $\mathbb{P}^{+}_{A,B,C} \to \mathbb{P}^{+}_{A,C}$ and the square being a pullback, using Lemma 2.2.70.

The fact that copycat strategies are $\mathbb{P}^{+}$-innocent amounts to commutation of

$$\begin{tikzcd}
\varnothing \arrow[r, "\Pi"] \arrow[d, equal] & \mathbb{P}^{+}_{A} \arrow[r, "\Sigma"] & \mathbb{P}^{+}_{A,A} \\
\varnothing \arrow[r, "\Pi"'] & \mathbb{P}_{A} \arrow[r, "\Sigma"'] \arrow[u, "\Delta"] & \mathbb{P}_{A,A} \arrow[u, "\Delta"']
\end{tikzcd}$$

up to isomorphism. The left-hand square is exact by Lemma 2.2.71 and the right-hand one is by Lemma 2.2.70, so the diagram commutes. This also shows that $\mathbb{P}^+$-copycat strategies are all copycat strategies, so they are exactly the units of composition. □

Of course, we have:

Proposition 6.4.15. *Both hypotheses are satisfied in the case of HON games with P-visibility, with $\mathbb{P}^+$ being Tsukada and Ong's games.*

Proof. This is simply because morphisms in HON games are exactly morphisms in Tsukada and Ong's games that are also prefixes. □

We will see in the next section why this notion of $\mathbb{P}^+$-innocence indeed makes sense for HON games by showing that boolean $\mathbb{P}^+$-innocent strategies are exactly traditional innocent strategies of HON games.

6.4.3 Boolean Innocence

Required: 6.4.2.
Recommended: ∅.

We finally consider boolean innocence. Let us start by showing that the notion of $\mathbb{P}^+$-innocence indeed makes sense, since boolean $\mathbb{P}^+$-innocent strategies correspond to traditional innocent strategies in HON games (where boolean $\mathbb{P}^+$-innocence and copycats are the obvious boolean counterparts to $\mathbb{P}^+$-innocence and copycats).

Proposition 6.4.16. *A strategy in the standard HON sense is innocent if and only if it is boolean $\mathbb{P}^+$-innocent for Tsukada and Ong's $\mathbb{P}^+$. It is copycat if and only if it is boolean $\mathbb{P}^+$-copycat.*

Proof. A boolean $\mathbb{P}^+$-innocent strategy is (isomorphic to) the image of a behaviour B in $\mathbb{V}_{A,B}$ through $\overline{\Delta_{\mathsf{k}_{A,B}}} \circ \overline{\prod_{\mathsf{i}_{A,B}}}$. Let us compute this on a play p:

$$\begin{aligned}
\overline{\Delta_{\mathsf{k}_{A,B}}}(\overline{\textstyle\prod_{\mathsf{i}_{A,B}}}(B))(p) &= \overline{\textstyle\prod_{\mathsf{i}_{A,B}}}(B)(\mathsf{k}_{A,B}(p)) \\
&\cong \int_{v \in \mathbb{V}_{A,B}} B(v)^{[\mathsf{i}_{A,B}(v), \mathsf{k}_{A,B}(p)]} \\
&\cong \bigwedge_{v \in \mathbb{V}_{A,B}} B(v)^{[\mathsf{i}_{A,B}(v), \mathsf{k}_{A,B}(p)]} && \text{because } 2 \text{ is an order} \\
&\cong \bigwedge_{f : \mathsf{i}_{A,B}(v) \to \mathsf{k}_{A,B}(p)} B(v) && \text{because } 0^{\varnothing} = 1,
\end{aligned}$$

which means that the strategy accepts p if and only if it accepts all of the views v that map into p, which is exactly the traditional definition of an innocent strategy. Conversely, any innocent strategy is isomorphic to $\overline{\Delta_{\mathsf{k}_{A,B}}} \circ \overline{\prod_{\mathsf{i}_{A,B}}} \circ \overline{\Delta_{\mathsf{j}_{A,B}}}$ of itself (where $\mathsf{j}_{A,B} : \mathbb{V}_{A,B} \to \mathbb{P}_{A,B}$ is the functor such that $\mathsf{i}_{A,B} = \mathsf{k}_{A,B} \mathsf{j}_{A,B}$) because this functor does not change the value of presheaves on views, and the result accepts a play if and only if it accepts all of its views, just like the strategy we started from. The case of copycats is similar. □

As mentioned before, innocent boolean strategies are not closed under composition. One usually either imposes a further determinism constraint, or relaxes the innocence constraint. It might be instructive to see how trying to derive the boolean case from the set-based one using our methods directly points to the problem.

Indeed, suppose given any innocent game setting $(\mathbb{A}, \mathbb{P}, \mathbb{V}, \mathsf{i})$. We would like to show that two boolean polynomial functors, say

$$\overline{P_1}, \overline{P_2} \colon \widetilde{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \to \widetilde{\mathbb{P}_{A,C}}$$

coincide. Here $\overline{P_1}$ is right extension to plays followed by composition and $\overline{P_2}$ is the same, followed by innocentisation, i.e., restriction to views followed by right extension to plays. (Let us note in passing that the arguments we are going to give directly apply to the case where, instead of an innocent game setting $(\mathbb{A}, \mathbb{P}, \mathbb{V}, \mathsf{i})$, we have an innocent game setting equipped with a subcategory of plays $(\mathbb{A}, \mathbb{V} \xrightarrow{\mathsf{j}} \mathbb{P} \xrightarrow{\mathsf{k}} \mathbb{P}^+)$ and study $\mathbb{P}^+$-innocent boolean strategies.) A first attempt to show this would be to show that

$$\begin{tikzcd}
\widehat{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "P_i"] \arrow[d, "\mathsf{l}_!"'] & \widehat{\mathbb{P}_{A,C}} \arrow[d, "\mathsf{l}_!"] \\
\widetilde{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "\overline{P_i}"'] & \widetilde{\mathbb{P}_{A,C}}
\end{tikzcd}$$

commutes (where, remember, $\mathsf{l}_!$ is as in Proposition 6.2.20), like when we deduced from associativity and unitality of composition of set-based strategies that composition of boolean strategies is also associative and unital. However, this does not hold because $\prod_{\mathsf{i}_{A,B}}$ is not in the class of functors that commute with $\mathsf{l}_!$ in general.

A second attempt is to show that each $\overline{P_i}$ factors as

$$\begin{tikzcd}
\widehat{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "P_i"] & \widehat{\mathbb{P}_{A,C}} \arrow[d, "\mathsf{l}_!"] \\
\widetilde{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "\overline{P_i}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}}.
\end{tikzcd}$$

Then because we have already shown that $P_1 \cong P_2$, we would automatically get $\overline{P_1} \cong \overline{P_2}$ as desired.

Now, both functors have the form

$$\widetilde{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \xrightarrow{\overline{Q}} \widetilde{\mathbb{P}_{A,B,C}} \xrightarrow{\overline{\Sigma_{\delta_1}}} \widetilde{\mathbb{P}_{A,C}} \xrightarrow{\overline{R}} \widetilde{\mathbb{P}_{A,C}},$$

where $\overline{Q}$ and $\overline{R}$ are only composed of $\overline{\Delta}$s and $\overline{\prod}$s. One can show that $\prod$s and Δs commute with $\mathsf{r}_!$, so we may hope to prove the desired factorisation like so:

$$\begin{tikzcd}
\widehat{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "Q"] & \widehat{\mathbb{P}_{A,B,C}} \arrow[r, "\Sigma_{\delta_1}"] \arrow[dr, phantom, "?"] & \widehat{\mathbb{P}_{A,C}} \arrow[r, "R"] & \widehat{\mathbb{P}_{A,C}} \arrow[dr, "\mathsf{l}_!"] \arrow[dr, Rightarrow, shift right=3, shorten=4mm, "\wr"'] & \\
\widetilde{\mathbb{V}_{A,B} + \mathbb{V}_{B,C}} \arrow[r, "\overline{Q}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,B,C}} \arrow[r, "\overline{\Sigma_{\delta_1}}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}} \arrow[r, "\overline{R}"'] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}} \arrow[r, equal] \arrow[u, "\mathsf{r}_!"] & \widetilde{\mathbb{P}_{A,C}}
\end{tikzcd}$$

where the right triangle commutes up to isomorphism because $\mathsf{l}_! \dashv \mathsf{r}_!$ is a reflection. (We could try other factorisations, by changing where we insert the counit of $\mathsf{l}_! \dashv \mathsf{r}_!$, but for $\overline{P_2}$, both Q and R contain a $\prod$ functor, which means that they cannot commute to $\mathsf{l}_!$.) The only problematic square is the marked one. And indeed, if the considered boolean strategies, say X_1 and X_2, are non-deterministic, then $\sum_{\delta_1}(\mathsf{r}_!(\overline{Q}[X_1, X_2]))$ may accept some $p \in \mathbb{P}_{A,C}$ in more than one way (see [46, 3.7.2]), which readily makes it non-isomorphic to any presheaf in the image of $\mathsf{r}_!$.

Remark. *Exactly the same argument explains why boolean composition cannot agree with set-based composition in general (as was noted in Remark 6.2.4).*

Standardly, the problem is overcome by restricting to deterministic strategies, for which the marked square commutes.

Finally, we proceed similarly in the case of copycat strategies. In this case, the problematic square

$$\begin{tikzcd}
\widehat{\mathbb{P}_A} \arrow[r, "\Sigma_{\iota_0}"] & \widehat{\mathbb{P}_{A,A}} \\
\widetilde{\mathbb{P}_A} \arrow[u, "\mathsf{r}_!"] \arrow[r, "\overline{\Sigma_{\iota_0}}"'] & \widetilde{\mathbb{P}_{A,A}} \arrow[u, "\mathsf{r}_!"']
\end{tikzcd}$$

does commute, because the involved colimits are coproducts which are either empty or singleton:

Proposition 6.4.17. *In any innocent game setting* $(\mathbb{A}, \mathbb{P}, \mathbb{V}, \mathsf{i})$*, boolean copycat strategies are innocent. This extends to the setting of Proposition 6.4.14, so that, as is standard, copycats are innocent* pb*-strategies.*

Proof. For any behaviour B and play p, because ι_0 is a discrete fibration, we have $\sum_{\iota_0}(\mathsf{r}_!(B))(p) \cong \sum_{\iota_0(s)=p} r(B(s))$ by Lemma 2.2.27, which is non-empty if and only if there is an s in $\mathbb{P}_A$ such that $\iota_0(s) = p$ and $B(s) = 1$. We also have $\mathsf{r}_!\left(\overline{\sum_{\iota_0}(B)}\right)(p) \cong r\left(\int^s B(s) \cdot [p, \iota_0(s)]\right)$, which is non-empty if and only if there is an s in $\mathbb{P}_A$ equipped with a morphism $f: p \to \iota_0(s)$ and such that $B(s) = 1$. By discrete fibredness of ι_0, both conditions above are equivalent. Therefore, commutation of the diagram above reduces to showing that $\sum_{\iota_0}(\mathsf{r}_!(B))(p)$ is either empty or a singleton, which is direct from the formula given above. □

6.5 Perspectives

We have introduced game settings and their innocent variant, a categorical framework for game semantics, aiming at unifying existing game models and facilitating the construction of new ones. Under mild hypotheses, we get categories of games and strategies in an abstract way, and, under further hypotheses, a category of games and innocent strategies. We have shown how this may be extended to categories with fewer morphisms and to boolean strategies, and how composition in all these different settings are related. We have shown that many traditional and recent models fit in this setting, and also exhibited some limitations of game settings (games where composition is not associative and composition of boolean innocent strategies).

A lot remains to be done, starting with the incorporation of further instances. The saturated view of AJM games (Section 6.3.3) seems at hand, but will involve significantly more advanced category theory, as Street fibrations and stacks will replace discrete fibrations and sheaves. We would first need to generalise the lemmas of exactness from fibrations to Street fibrations, which should prove very easy, but the tougher part would be to prove that local pushforward squares (with a Street fibration) are distributive. Indeed, our current method uses the adjoint equivalence between presheaves and discrete fibrations, and it is not clear whether this would work with Street fibrations.

A less obvious direction is the treatment of more exotic game models [5, 80, 81, 84, 83, 47, 92, 21], notably those based on event structures. It also seems that strategies may compose in our game model (defined in Chapter 5), based on the proof of Proposition 6.4.7. So maybe our game model also fits this framework.

Another direction is categorification: instead of reasoning up to isomorphism, we could refine our point of view and prove that games and strategies in fact form a bicategory, as, e.g., in [92].

We also plan to go beyond mere categories of games and strategies and construct structured categories of various kinds, depending on the considered language. These could be, e.g., cartesian closed, symmetric monoidal closed, linear, or Freyd categories. Indeed, proving that we have a category of games and strategies is actually a weak result if we want to interpret any serious language in it. This brings us to another research direction: beyond game models, we should investigate game semantics, i.e., the correspondence with operational semantics, as initiated in [27] in a different setting.

Beyond the realm of game semantics itself, we have the following formula for composition:

$$\begin{aligned}(\sigma;\tau)(p) &= \textstyle\sum_{\delta_1}(\prod_{\nabla}(\Delta_{\delta_2+\delta_0}([\sigma,\tau])))(p)\\ &\cong \int^{u} \textstyle\prod_{\nabla}(\Delta_{\delta_2+\delta_0([\sigma,\tau])})(u) \times [p,\delta_1 u]\\ &\cong \int^{u} \sigma(\delta_2(u)) \times \tau(\delta_0(u)) \times [p,\delta_1 u],\end{aligned}$$

which looks awfully like Day convolution. Indeed, it seems like this could be written as

$$(\sigma;\tau)(p) \cong \int^{(p_1,p_2)\in\mathbb{P}_{A,B}\times\mathbb{P}_{B,C}} \sigma(p_1) \times \tau(p_2) \times \mathbb{P}^{op}_{A,C}(p_1 \otimes p_2, p)$$

where $\otimes$ is a kind of object that represents the hiding of p_1 and p_2 along B if they coincide, and some "error" element otherwise. The idea is that, in some cases (e.g., simple games), we have that $\mathbb{P}_{A,B,C} \cong \mathbb{P}_{A,B} \times_{\mathbb{P}_B} \mathbb{P}_{B,C}$, so each interaction sequence u is actually given by a pair (p_1,p_2) with p_1 in $\mathbb{P}_{A,B}$ and p_2 in $\mathbb{P}_{B,C}$, in which case the formula above makes sense. For it to make sense in general seems to require a slight generalisation of Day convolution from the context of monoidal categories to a broader one: indeed, we want to take the objects p_1 and p_2 in different categories, and the tensorial product lands in yet another category.

Bibliography

[1] LICS 2015. *Proc. 30th Symposium on Logic in Computer Science*, 2015. IEEE.

[2] Samson Abramsky. Sequentiality vs. concurrency in games and logic. *Mathematical Structures in Computer Science*, 13(4):531–565, 2003. doi: 10.1017/S0960129503003980. URL `https://doi.org/10.1017/S0960129503003980`.

[3] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.

[4] Samson Abramsky and Radha Jagadeesan. A game semantics for generic polymorphism. *Annals of Pure and Applied Logic*, 133(1-3):3–37, 2005.

[5] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *LICS*, pages 431–442. IEEE, 1999. doi: 10.1109/LICS.1999.782638.

[6] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000. doi: 10.1006/inco.2000.2930.

[7] Samson Abramsky, Dan R. Ghica, Andrzej S. Murawski, C.-H. Luke Ong, and Ian D. B. Stark. Nominal games and full abstraction for the nu-calculus. In *Logic in Computer Science, 2004. Proceedings of the 19th Annual IEEE Symposium on*, pages 150–159. IEEE, 2004.

[8] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992. doi: 10.1093/logcom/2.3.297. URL `http://dx.doi.org/10.1093/logcom/2.3.297`.

[9] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. Believe it or not, AJM's Games Model is a Model of Classical Linear Logic. In *LICS*, pages 68–75. IEEE, 1997. ISBN 0-8186-7925-5. doi: 10.1109/LICS.1997.614933.

[10] Paolo Baldan, Andrea Corradini, Tobias Heindel, Barbara König, and Paweł Sobociński. Processes and unfoldings: concurrent computations in adhesive categories. *Mathematical Structures in Computer Science*, 24(4), 2014. doi: 10.1017/S096012951200031X.

[11] Henk Barendregt. The lambda calculus: its syntax and semantics. *Studies in logic and the foundations of Mathematics*, 1984.

[12] Gérard Berry and Gérard Boudol. The chemical abstract machine. In *Proc. 17th International Symposium on Principles of Programming Languages*, pages 81–94, 1990. doi: 10.1145/96709.96717.

[13] Gérard Berry and Pierre-Louis Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3):265–321, 1982.

[14] Andreas Blass. Degrees of indeterminacy in games. *Fundamenta Mathematica*, LXXVII:151–162, 1972.

[15] Andreas Blass. A game semantics for linear logic. *Annals of Pure and Applied Logic*, 56(1–3):183–220, 1992. doi: 10.1016/0168-0072(92)90073-9.

[16] Pierre Boudes. Thick subtrees, games and experiments. In *TLCA*, volume 5608 of *LNCS*, pages 65–79. Springer, 2009. doi: 10.1007/978-3-642-02273-9_7.

[17] Aldridge K. Bousfield. Constructions of factorization systems in categories. *Journal of Pure and Applied Algebra*, 9(2-3):287–329, 1977.

[18] Nathan J. Bowler. *A unified approach to the construction of categories of games*. PhD thesis, University of Cambridge, 2011.

[19] Aurelio Carboni, Stephen Lack, and Robert F. C. Walters. Introduction to extensive and distributive categories. *Journal of Pure and Applied Algebra*, 84(2), 1993.

[20] Simon Castellan, Pierre Clairambault, and Glynn Winskel. Concurrent Hyland-Ong games. GaLoP, 2014.

[21] Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of PCF. In LICS 2015 [1].

[22] Pierre Clairambault. Strong functors and interleaving fixpoints in game semantics. *RAIRO - Theor. Inf. and Applic.*, 47(1):25–68, 2013. doi: 10.1051/ita/2012028. URL `https://doi.org/10.1051/ita/2012028`.

[23] Thierry Coquand. A semantics of evidence for classical arithmetic. *Journal of Symbolic Logic*, 60(1):325–337, 1995. doi: 10.2307/2275524.

[24] Clovis Eberhart and Tom Hirschowitz. Presheaves for Processes and Unfoldings. `http://www.lama.univ-savoie.fr/~eberhart/ProcUnfold.pdf`, 2015. Online extended abstract for a talk at CALCO Early Ideas.

[25] Clovis Eberhart and Tom Hirschowitz. Justified sequences in string diagrams: a comparison between two approaches to concurrent game semantics. Preprint, 2016. URL `https://hal.archives-ouvertes.fr/hal-01372582`.

[26] Clovis Eberhart and Tom Hirschowitz. Justified sequences in string diagrams: a comparison between two approaches to concurrent game semantics. In *Proc. 7th International Conference on Algebra and Coalgebra in Computer Science*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017. To appear.

[27] Clovis Eberhart and Tom Hirschowitz. Game semantics as a singular functor, and definability as geometric realisation. Preprint, 2017. URL `https://hal.archives-ouvertes.fr/hal-01527171`.

[28] Clovis Eberhart and Tom Hirschowitz. What's in a game? A theory of game models. Accepted for publication at LICS 2018, November 2017. URL `https://hal.archives-ouvertes.fr/hal-01634162`.

[29] Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. An intensionally fully-abstract sheaf model for π. In *Proc. 6th International Conference on Algebra and Coalgebra in Computer Science*, volume 35 of *LIPIcs*, pages 86–100. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. doi: 10.4230/LIPIcs.CALCO.2015.86.

[30] Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. An intensionally fully-abstract sheaf model for π (expanded version). *Logical Methods in Computer Science*, 13(4), 2017. doi: 10.23638/LMCS-13(4:9)2017. URL `https://doi.org/10.23638/LMCS-13(4:9)2017`.

[31] Charles Ehresmann. Catégories structurées. *Annales scientifiques de l'Ecole Normale Supérieure*, 80(4):349–426, 1963.

[32] Charles Ehresmann. *Catégories et structures*. Dunod, 1965.

[33] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *LNCS*. Springer, 1979.

[34] Hartmut Ehrig, Reiko Heckel, Martin Korff, Michael Löwe, Leila Ribeiro, Annika Wagner, and Andrea Corradini. Algebraic approaches to graph transformation–part II: Single pushout approach and comparison with double pushout approach. In *Handbook Of Graph Grammars And Computing By Graph Transformation: Volume 1: Foundations*, pages 247–312. World Scientific, 1997.

[35] Marcelo P. Fiore. Discrete generalised polynomial functors - (extended abstract). In *Proc. 39th International Colloquium on Automata, Languages and Programming*, volume 7392 of *LNCS*, pages 214–226. Springer, 2012. doi: 10.1007/978-3-642-31585-5_22.

[36] Cédric Fournet and Georges Gonthier. The reflexive cham and the join-calculus. In *Proc. 23rd International Symposium on Principles of Programming Languages*, pages 372–385, 1996. ISBN 0-89791-769-3. doi: 10.1145/237721.237805.

[37] Richard H. G. Garner. *Polycategories*. PhD thesis, University of Cambridge, 2006.

[38] Richard H. G. Garner. A homotopy-theoretic universal property of Leinster's operad for weak ω-categories. *Math. Proc. Camb. Phil. Soc.*, 147: 615–628, 2009.

[39] Richard H. G. Garner and Tom Hirschowitz. Shapely monads and analytic functors. *Journal of Logic and Computation*, 28(1):33–83, 2018. doi: 10.1093/logcom/exx029. URL `http://dx.doi.org/10.1093/logcom/exx029`.

[40] Dan R. Ghica and Andrzej S. Murawski. Angelic semantics of fine-grained concurrency. *Annals of Pure and Applied Logic*, 151(2–3):89 – 114, 2008. doi: 10.1016/j.apal.2007.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0168007207000851`.

[41] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[42] Marco Grandis and Robert Paré. Limits in double categories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 40(3):162–220, 1999.

[43] Marco Grandis and Robert Paré. Adjoints for double categories. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 45(3):193–240, 2004.

[44] Marco Grandis and Robert Paré. Kan extensions in double categories (On weak double categories, Part III). *Theory and Applications of Categories*, 20(8):152–185, 2008.

[45] René Guitart. Relations et carrés exacts. *Annales des Sciences Mathématiques du Québec*, 4(2):103–125, 1980.

[46] Russell Harmer. *Games and Full Abstraction for Nondeterministic Languages*. PhD thesis, Imperial College, University of London, 1999.

[47] Russell Harmer, J. Martin E. Hyland, and Paul-André Melliès. Categorical combinatorics for innocent strategies. In *Proc. 22nd Symposium on Logic in Computer Science*, pages 379–388. IEEE, 2007. doi: 10.1109/LICS.2007.14.

[48] Michel Hirschowitz, André Hirschowitz, and Tom Hirschowitz. A theory for game theories. In *Proc. 27th Foundations of Software Technology and Theoretical Computer Science*, pages 192–203. Springer, 2007.

[49] Tom Hirschowitz. Full abstraction for fair testing in CCS. In *Proc. 5th International Conference on Algebra and Coalgebra in Computer Science*, volume 8089 of *LNCS*, pages 175–190. Springer, 2013. doi: 10.1007/978-3-642-40206-7_14.

[50] Tom Hirschowitz. Full abstraction for fair testing in CCS (expanded version). *Logical Methods in Computer Science*, 10(4), 2014. doi: 10.2168/LMCS-10(4:2)2014.

[51] Tom Hirschowitz and Damien Pous. Innocent strategies as presheaves and interactive equivalences for CCS. In *Proc. of Interaction and Concurrency Experience*, pages 2–24. Electronic Proceedings in Theoretical Computer Science, 2011. doi: 10.4204/EPTCS.59.2.

[52] Kohei Honda and Nobuko Yoshida. Game-Theoretic Analysis of Call-by-Value Computation. *tcs*, 221(1-2):393–456, 1999. doi: 10.1016/S0304-3975(99)00039-0. URL `https://doi.org/10.1016/S0304-3975(99)00039-0`.

[53] Dominic J. D. Hughes. Games and definability for System F. In *Logic in Computer Science, 1997. LICS'97. Proceedings., 12th Annual IEEE Symposium on*, pages 76–86. IEEE, 1997.

[54] J. Martin E. Hyland. Game semantics. *Semantics and logics of computation*, 14:131, 1997.

[55] J. Martin E. Hyland. Game semantics. In Andrew M. Pitts and Peter Dybjer, editors, *Semantics and Logics of Computation*, pages 131–184. Cambridge University Press, 1997.

[56] J. Martin E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000. doi: 10.1006/inco.2000.2917.

[57] Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium - Volume 2*. Oxford University Press, 2002.

[58] André Joyal. Remarques sur la théorie des jeux à deux personnes. *Gazette des Sciences Mathématiques du Québec*, 1(4):46–52, 1977.

[59] André Joyal. Foncteurs analytiques et espèces de structure. In *Combinatoire énumérative (Montréal 1985)*, volume 1234 of *Lecture Notes in Mathematics*, pages 126–159. Springer, 1986.

[60] G. Maxwell Kelly. *Basic concepts of enriched category theory*, volume 64 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1982. Republished as: *Reprints in Theory and Applications of Categories 10* (2005).

[61] G. Maxwell Kelly. Elementary observations on 2-categorical limits. *Bulletin of the Australian Mathematical Society*, 39:301–317, 1989.

[62] Joachim Kock. Notes on polynomial functors. Working notes, 2009. URL `http://mat.uab.es/~kock/cat/polynomial.pdf`.

[63] Jean-Louis Krivine. A call-by-name lambda-calculus machine. *Higher-Order and Symbolic Computation*, 20(3):199–207, 2007. ISSN 1573-0557. doi: 10.1007/s10990-007-9018-9. URL `https://doi.org/10.1007/s10990-007-9018-9`.

[64] Stephen Lack and Paweł Sobociński. Adhesive categories. In *Proc. 7th Foundations of Software Science and Computational Structures*, volume 2987 of *LNCS*, pages 273–288. Springer, 2004. doi: 10.1007/978-3-540-24727-2_20.

[65] Yves Lafont. Interaction combinators. *iandcomp*, 137(1):69–101, 1997. doi: 10.1006/inco.1997.2643. URL `https://doi.org/10.1006/inco.1997.2643`.

[66] James Laird. A categorical semantics of higher order store. *Electronic notes in Theoretical Computer Science*, 69:209–226, 2003.

[67] James Laird. A game semantics of the asynchronous pi-calculus. In *Proc. 16th International Conference on Concurrency Theory*, volume 3653 of *LNCS*, pages 51–65. Springer, 2005. doi: 10.1007/11539452_8.

[68] James Laird. Game semantics and linear CPS interpretation. *Theoretical computer science*, 333(1-2):199–224, 2005.

[69] James Laird. Game semantics for higher-order concurrency. In *Proc. 26th Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *LNCS*, pages 417–428. Springer, 2006. doi: 10.1007/11944836_38.

[70] James Laird. Game semantics for call-by-value polymorphism. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming*, pages 187–198, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-14162-1.

[71] James Laird. Game Semantics for a Polymorphic Programming Language. *J. ACM*, 60(4):29:1–29:27, September 2013. ISSN 0004-5411. doi: 10.1145/2508028.2505986. URL `http://doi.acm.org/10.1145/2508028.2505986`.

[72] Tom Leinster. *Higher Operads, Higher Categories*, volume 298 of *London Mathematical Society Lecture Notes.* Cambridge University Press, Cambridge, 2004.

[73] Paul Blain Levy. Morphisms between plays. GaLoP, 2013.

[74] Ralph Loader. Finitary pcf is not decidable. *Theoretical Computer Science*, 266(1-2):341–364, 2001.

[75] Paul Lorenzen and Kuno Lorenz. *Dialogische logik.* Wissenschaftliche Buchgesellschaft, 1978.

[76] Saunders Mac Lane. *Categories for the Working Mathematician.* Number 5 in Graduate Texts in Mathematics. Springer, 2nd edition, 1998.

[77] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory.* Universitext. Springer, 1992.

[78] Guy McCusker. *Games and Full Abstraction for a Functional Metalanguage with Recursive Types.* PhD thesis, Imperial College, University of London, 1996.

[79] Paul-André Mellies. Asynchronous games 1: Uniformity by group invariance, 2003.

[80] Paul-André Melliès. Asynchronous games 2: the true concurrency of innocence. In *Proc. 15th International Conference on Concurrency Theory*, volume 3170 of *LNCS*, pages 448–465. Springer, 2004. doi: 10.1007/978-3-540-28644-8_29.

[81] Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. In *Proc. 20th Symposium on Logic in Computer Science*, pages 386–395. IEEE, 2005. ISBN 0-7695-2266-1. doi: 10.1109/LICS.2005.6.

[82] Paul-André Melliès. Asynchronous games 3: An innocent model of linear logic. *Electronic Notes in Theoretical Computer Science*, 122:171–192, 2005.

[83] Paul-André Melliès. Game semantics in string diagrams. In *Proc. 27th Symposium on Logic in Computer Science*, pages 481–490. IEEE, 2012. doi: 10.1109/LICS.2012.58.

[84] Paul-André Melliès and Samuel Mimram. Asynchronous games: Innocence without alternation. In *Proc. 19th International Conference on Concurrency Theory*, volume 4703 of *LNCS*, pages 395–411. Springer, 2007. doi: 10.1007/978-3-540-74407-8_27.

[85] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, I/II. *Information and Computation*, 100(1):1–77, 1992.

[86] Andrzej S. Murawski and Nikos Tzevelekos. Nominal game semantics. *Foundations and Trends in Programming Languages*, 2(4):191–269, 2016. doi: 10.1561/2500000017. URL `https://doi.org/10.1561/2500000017`.

[87] Hanno Nickau. Hereditarily sequential functionals. In *Proc. Logical Foundations of Computer Science*, volume 813 of *LNCS*, pages 253–264. Springer, 1994. doi: 10.1007/3-540-58140-5_25.

[88] Mogens Nielsen, Gordon Plotkin, and Glynn Winskel. Event structures and domains, part 1. *Theoretical Computer Science*, 13:65–108, 1981.

[89] Robert Paré. Universal covering categories. *Rendiconti dell'Istituto di Matematica dell'Università di Trieste*, 1993.

[90] Robert Paré. Yoneda theory for double categories. *Theory and Applications of Categories*, 25(17):436–489, 2011.

[91] Michal R. Przybylek. The other pullback lemma. *ArXiv e-prints*, November 2013.

[92] Silvain Rideau and Glynn Winskel. Concurrent strategies. In *Proc. 26th Symposium on Logic in Computer Science*, pages 409–418. IEEE, 2011. doi: 10.1109/LICS.2011.13.

[93] Dana S. Scott. Domains for denotational semantics. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming*, pages 577–610, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN 978-3-540-39308-5.

[94] Michael Shulman. Framed bicategories and monoidal fibrations. *Theory and Applications of Categories*, 13(9):147–163, 2008.

[95] Morten Heine Sørensen and Pawel Urzyczyn. *Lectures on the Curry-Howard isomorphism*, volume 149. Elsevier, 2006.

[96] Takeshi Tsukada and C.-H. Luke Ong. Innocent strategies are sheaves over plays - deterministic, non-deterministic and probabilistic innocence. *CoRR*, abs/1409.2764, 2014. URL `http://arxiv.org/abs/1409.2764`.

[97] Takeshi Tsukada and C.-H. Luke Ong. Nondeterminism in game semantics via sheaves. In LICS 2015 [1].

[98] Mark Weber. Generic morphisms, parametric representations and weakly cartesian monads. *Theory and Applications of Categories*, 13:191–234, 2004.

[99] Mark Weber. Familial 2-functors and parametric right adjoints. *Theory and Applications of Categories*, 18(22):665–732, 2007.

Appendix A

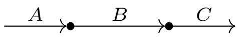
A Proof of View-Analyticity in Tsukada and Ong's Model

This appendix exposes the original proof of view-analyticity in Tsukada and Ong's model, as we proved it. It requires tools from Chapter 5, and while it is arguably more complex than Takeshi Tsukada's proof, we have chosen to keep it inside this dissertation because it shows that we can define a notion of hiding on plays defined as string diagrams for the λ-calculus.

We first need to introduce a notion of interaction sequence in the setting of string diagrams, which we will simply call an *interaction.* Let $(A \vdash B \vdash C)$ be the position given by the pushout:

$$\begin{tikzcd}
B \arrow[r, "t"] \arrow[d, "s_1"'] & (A \vdash B) \arrow[d] \\
(B \vdash C) \arrow[r] \arrow[ur, phantom, "\lrcorner", very near end] & (A \vdash B \vdash C).
\end{tikzcd}$$

If we draw this position, we get:



In any position reached by a play starting from $(A \vdash B \vdash C)$, players can be split in two groups: the avatars of the left-hand player and those of the right-hand one. Similarly, edges can be split into three groups, depending on which edge they stem from.

Definition A.0.1. *Let $\mathbb{I}(A, B, C)$ denote the full subcategory of $\mathbb{E}(A \vdash B \vdash C)$ spanning plays whose "internal" moves (those that are played on an edge stemming from the middle one) are β moves. We call such plays* interactions.

Let us show on an example how interactions on (A, B, C) can be mapped to plays on (A, C). Let us consider the interaction on the left of Figure A.1, and its mapping to a play on (A, C) on the right (written with the initial position at the top for readability). The interaction of Figure A.1 corresponds to the interaction sequence:

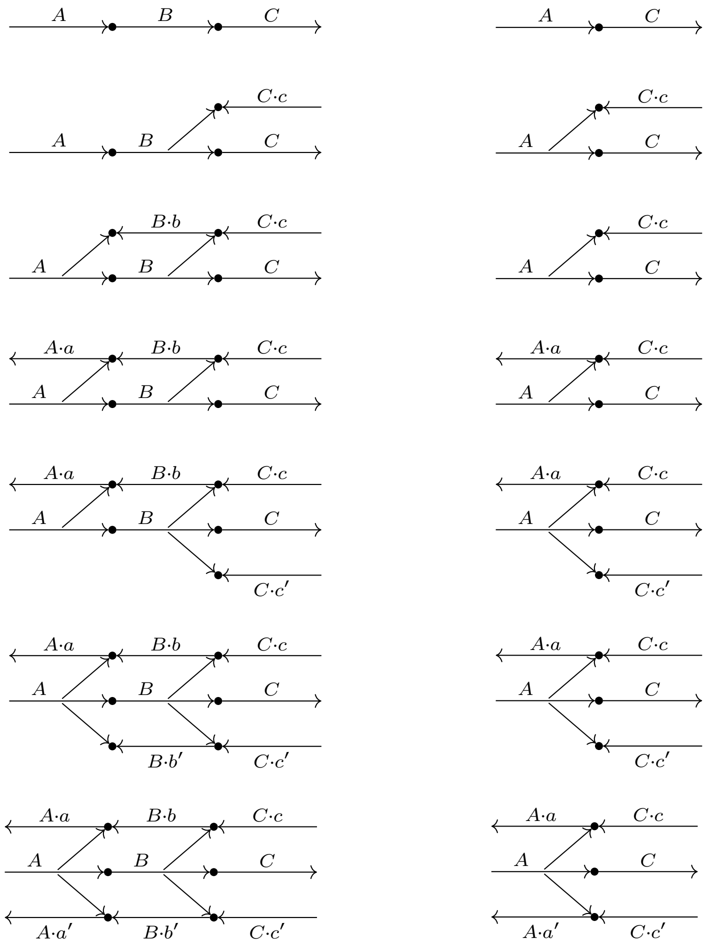


Figure A.1: An interaction and its projection

$$\begin{tikzcd}
A \arrow[r] & B \arrow[r] & C \\
& & c \\
& b \arrow[ur] & \\
a \arrow[ur] & & \\
& & c' \\
& b' \arrow[ur] & \\
a'. \arrow[ur] & &
\end{tikzcd}$$

One can map the interaction on the left to the play on the right by playing only Λ and @ moves. For this to make sense, we have to show that any move that is played in the interaction may be played in the play created by playing only Λ and @ moves.

To prove this, the idea is to take a linearisation of our play U and to draw *zones* in every intermediate position X_i. Each zone in X_i will correspond to a player in the projection of X_i to $(A \vdash C)$. They are defined inductively by:

- in the initial position X_0, there is only one zone and it encompasses the whole position,
- after an @ move, zones "do not change", in the sense that the newly created player is incorporated to the zones of the player who played the @ move,
- after a β move, all zones are left unaltered except for the (only) zone that contains the edge the β move is played on, which is extended with the newly created player,
- after a Λ move, all zones are left unchanged, and a new zone is created, containing all players that are in the (unique) zone of the player who played the Λ move, plus the newly created player.

To prove that this actually works requires a few lemmas that are simply proved by mutual induction on the number of moves in a play (this is purely technical and not difficult).

In the drawing of Figure A.1, there are three zones in the interaction's bottom-most position: one that contains the top two players and the left-side one, one that contains the bottom two players and the left-side one, and one that contains the left-side and right-side players.

Since a single zone is created when a Λ move is played and zones are left unchanged in the other cases, we can see that zones correspond to players in the projection. More precisely, zones even have the same "interface" as players (incoming and outgoing edges), except that zones may have the same incoming edge multiple times in their interfaces: @ moves add an outgoing edge both to a player and an interface, Λ moves create both a player and an interface with the outgoing edge replaced by an incoming one, and β moves do not change the shape of the interface (except that it may duplicate some incoming edges). This shows that all moves that are played in the interaction can also be played in the projection on (A, C), which is thus well defined.

Now, we want to take a zone Z of X_i and define its *history*, i.e., the "smallest" subplay of U that starts from its initial position X_0 and contains Z. The history is defined recursively as:

- the empty play if Z is the only zone of X_0,
- if Z is a zone of X_{n+1}, then its history is the history of the zone in X_n it comes from, plus the move played between X_n and X_{n+1} if the created player is in Z (i.e., take all moves that happen in Z, except if they are Λ moves and Z is the zone where the initial player is).

To prove Proposition 6.4.7, we will take a well-chosen zone Z, define W to be its history, and show that W corresponds to an interaction sequence $w_{A,B,C}$ that satisfies all the properties to show that Tsukada and Ong's game setting is view-analytic.

Proof sketch of Proposition 6.4.7. Consider an interaction sequence $u_{A,B,C}$, a view $v_{A,B}$, and a morphism $f\colon v_{A,B} \to \delta_2(u_{A,B,C})$ (the construction is the same for $\delta_0(u_{A,B,C})$). Just like there is a full embedding from plays as justified sequences to plays as string diagrams, there is a full embedding of interaction sequences into interactions, which takes all moves of $u_{A,B,C}$ and plays a Λ move for each Opponent move from the point of view of the interaction sequence (i.e., Opponent moves in C and Proponent moves in A), @ moves for Proponent moves, and β moves for internal moves on B. Let U denote the interaction that corresponds to $u_{A,B,C}$ through this mapping. Now, each move of $u_{A,B,C}$ corresponds to a move of U, so, in particular, the last move of $f(v_{A,B})$ corresponds to some move in U. That move can either be an Opponent move in A or a Proponent move in B, which means that it corresponds to either an @ move or a β move. In particular, it corresponds to a zone of U: the zone the move is played on (were the move a Λ, it would be slightly more difficult because it might correspond to two different zones, but this case never happens). We define W to be the history of that zone, plus all the moves that happen in that zone until the next @ move (including all these extra moves is needed to ensure that W corresponds to a play that ends in A or C).

Now we want to show that: (i) W corresponds to an interaction sequence $w_{A,B,C}$ on (A, B, C) through the mapping defined above, (ii) f canonically factors through $\delta_2(w_{A,B,C})$, and (iii) $\delta_1(w_{A,B,C})$ is a view, which will prove view-analyticity.

For the first point, W corresponds to an interaction sequence if and only if it ends in A or C and its obvious left and right projections correspond to plays as justified sequences. The first point is by definition of W. For the projections, we know that they correspond to P-visible justified sequences (because all string diagrammatic plays do), so we just need to show that they are alternating and of even length. Because U is built from $u_{A,B,C}$, by the switching condition, we know that there is at most one positive player in any of the X_n's, so whenever a positive player is created, they must play the next move (or at least be part of it if the next move is a β). Now, if we look at one of the projections of U, say the one to (A, B), if a Λ move happens in the projection of W (such a move may be the projection of a β move of U), then it creates a positive player in the zone, which will necessarily play the next move in U. This next move must therefore also be in W. If the Λ move is not in W, then a positive player is indeed created, but it is not in the zone, so the move it plays will not be in W either. Therefore, W always takes two moves of U of opposite polarities in a row, so it is alternating and of even length.

For the second point, we have a morphism $h{:}\, w_{A,B,C} \to u_{A,B,C}$ given by the fact that it is a sub-sequence of $u_{A,B,C}$ (more precisely, it is given by the inclusion of W into U and the fact that the mapping of interaction sequences into interactions is fully faithful). Moreover, all moves of $v_{A,B}$ are necessarily played in $w_{A,B,C}$ because W contains the final move of $f(v_{A,B})$, so it must contain the views of all involved players (by construction of string diagrams). Since $v_{A,B}$ is a view, it becomes a branch V when turned into a string diagram, and since we know where to map the final player into W, there is at most one map from V to the projection $W_{A,B}$ of W to (A, B). But that map must exist because W contains all the moves that made playing the last move of $v_{A,B}$ possible in U. Therefore, there is a unique morphism $V \to W_{A,B}$ that corresponds to the morphism $v_{A,B} \to \delta_2(u_{A,B,C})$. Since the mapping is fully faithful, this means that there is a morphism $g{:}\, v_{A,B} \to \delta_2(w_{A,B,C})$ (because projections of interaction sequences are compatible with projections of interactions). We have $f = \delta_2(h)g$.

Finally, for the last point, we need to show that the projection $\delta_1(w_{A,B,C})$

is a view, for which it is sufficient to show that the projection of W to (A, C) is a view. To show that a string diagram is a view, it suffices to show that it is non-empty and that the player who plays the next move is always the newly created player. It is obviously non-empty because W contains the last move of $f(v_{A,B})$ and ends in either A or C. Now, since the moves that are played in W are only the ones that create a new player in a determined zone and zones bijectively correspond to players in the projection to (A, C), moves of W "follow" the distinguished zone, which is either the expanded zone (in the case of β or @ moves) or the newly created one (in the case of Λ moves), so they follow the newly created player. □

Résumé

La programmation concurrente a grandement gagné en popularité ces dernières années, notamment grâce à l'augmentation du nombre de cœurs par processeur et à la mise en œuvre pratique de la programmation distribuée. Cependant, la programmation concurrente est bien moins intuitive que les paradigmes de programmation traditionnels, et la plupart des programmes reste donc incapable d'utiliser plusieurs cœurs en parallèle. Une façon de rendre la programmation concurrente plus intuitive pourrait être de créer des langages spécifiquement pour faire de la concurrence, mais cela nécessite de comprendre les notions qui sous-tendent la programmation concurrente. On peut faire un parallèle avec la programmation fonctionnelle : un aspect intéressant (pour un théoricien) d'OCaml ou de Haskell est que ce sont des langages dont le développement s'appuie sur une théorie bien comprise. On peut utiliser ces langages pour tester l'efficacité des techniques de programmation fonctionnelle sur de vrais problèmes, plutôt que sur des problèmes académiques. Si ces techniques s'avèrent utiles, les paradigmes de programmation fonctionnelle peuvent être ajoutés à des langages plus « grand public », tels que Python ou C++, où ils pourront être utilisés pour résoudre certains problèmes plus facilement. Ce travail est une contribution à la sémantique de jeux concurrents, un domaine de recherche qui utilise la sémantique de jeux pour mieux comprendre la concurrence.

Sémantique des langages de programmation

La *sémantique des langages de programmation* (que l'on appellera aussi simplement *sémantique*) est un domaine de l'informatique dont le but est de donner un *sens mathématique* aux programmes informatiques. En effet, un *terme* d'un langage n'est rien de plus qu'une suite de symboles, et elle n'a pas de sens en soi. Elle ne fait sens que dans le contexte d'un langage de programmation particulier. L'idée de la sémantique est de construire des modèles mathématiques des langages de programmation pour prouver des propriétés de ces langages.

Il y a plusieurs raisons pour lesquelles on peut vouloir donner un sens mathématique à un programme : prouver que les programmes écrits dans un certain langage ont de bonnes propriétés, prouver qu'un programme particulier à le comportement attendu, prouver qu'un programme s'arrête en un temps raisonnable, prouver que deux programmes ont le même comportement... Une autre raison d'étudier les langages de programmation réside dans les liens qu'ils entretiennent avec la logique. C'est ce qu'on appelle la *correspondance de Curry-Howard* [95]. Sous sa forme la plus basique, cette correspondance dit qu'un type A d'un langage de programmation peut aussi être vu comme une formule logique

$[\![A]\!]$ d'une certaine logique, et vice versa. Mais ce qui est intéressant dans cette correspondance est le point suivant : les programmes de type A correspondent à des preuves de $[\![A]\!]$ dans la logique. La correspondance va même plus loin, en disant que la composition des programmes correspond à des *coupures* dans la logique (une coupure est une étape dans une preuve où l'on ne prouve rien de nouveau, par exemple quand on introduit un lemme). Enfin, la correspondance est aussi dynamique, au sens où la *normalisation* (l'exécution) d'un programme correspond à l'*élimination des coupures* de la preuve (l'élimination des coupures est un processus qui transforme une preuve en une autre preuve de la même formule qui ne contient aucune coupure, et qui peut être vue comme le fait de remplacer tous les lemmes par leurs preuves chaque fois qu'ils sont utilisés, au lieu de prouver les lemmes une seule fois pour toutes). La sémantique est donc une façon de mieux comprendre la logique, et vice versa.

Une carte (surannée) de la sémantique

On va maintenant ébaucher une vue légèrement périmée de la sémantique (que nous nous efforcerons de corriger dans un second temps, en montrant que le paysage de la sémantique est plus complexe).

La sémantique peut prendre différentes formes, plus ou moins adaptées à démontrer certains types de propriétés. Il y a deux branches principales : la *sémantique opérationnelle*, qui décrit un programme comme une sorte de machine, et la *sémantique dénotationnelle*, qui décrit les programmes comme des structures mathématiques bien connues.

Sémantique opérationnelle

La sémantique opérationnelle est probablement la façon de représenter les programmes qui est la plus proche des intuitions qu'ont les programmeurs. Elle décrit les programmes comme des suites d'instructions exécutées par une sorte de machine. Cette description est très proche de ce qui se passe en effet dans un ordinateur, mais les ensembles d'instructions utilisés en sémantique opérationnelle simplifient les instructions réellement utilisées.

La sémantique opérationnelle revêt différentes formes, mais toutes se conforment au schéma esquissé ci-dessus. La forme de sémantique opérationnelle la plus commune s'appuie sur les *systèmes de transitions étiquetés* (ou LTS, pour *labelled transition system*), qui sont en gros des graphes dont les sommets sont les différents états possibles d'un programme et les arêtes sont les étapes d'exécution d'un programme qui commence dans un certain état et finit dans un autre. Donner les règles de réduction d'un langage formel (tel que le λ-calcul [11] ou le π-calcul [85]) correspond exactement à définir un LTS dont les sommets sont les termes du langage et les arêtes sont les réductions possibles. Voici, par exemple, un LTS très simple pour le λ-calcul :

$$\frac{}{(\lambda x.M)N \to M[N/x]} \qquad \frac{M \to M'}{MN \to M'N} \qquad \frac{N \to N'}{MN \to MN'}.$$

Certains LTS s'appuient sur la notion de *machine abstraite* [63]. Comme leur nom l'indique, les machines abstraites sont dans un certain sens encore plus proche de l'idée d'une machine qui exécute une suite d'instructions. Elles exécutent en général un programme (ou *terme*) en face d'un contexte appelé *pile*, et le terme et la pile peuvent tous deux être modifiés par les différentes instructions de la machine. Par exemple, la machine peut empiler les arguments d'une application puis les dépiler au moment de les utiliser. C'est exactement ce que fait cette machine pour le λ-calcul :

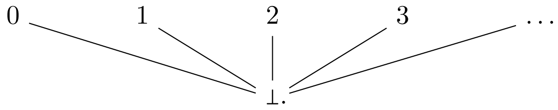
$$MN \star \pi \to M \star N :: \pi \qquad\qquad (\lambda x.M) \star N :: \pi \to M[N/x] \star \pi.$$

Ici, π désigne la pile (une suite de λ-termes). La première règle dit que, face à une application, la machine empile l'argument, et la seconde dit que, face à une abstraction, la machine dépile un argument et effectue la substitution.

Sémantique dénotationnelle

On a vu que, pour la sémantique opérationnelle, l'interprétation d'un programme est proche de sa syntaxe. À l'inverse, en sémantique dénotationnelle, les programmes sont représentés par des structures mathématiques connues. L'idée est d'interpréter un type A par un certain type d'espace $[\![A]\!]$ (par exemple, des espaces topologiques) et les programmes de type $A \to B$ par des morphismes de $[\![A]\!]$ dans $[\![B]\!]$ (dans le cas des espaces topologiques, les morphismes sont les fonctions continues).

Les *domaines de Scott* [93] offrent un exemple d'une telle sémantique. Grossièrement, un domaine de Scott est un ensemble muni d'un ordre qui représente de le niveau d'« information » que l'on possède sur un certain type d'objet : $x < y$ signifie que y donne plus d'informations sur l'objet qu'il représente que x. Par exemple, le domaine de Scott pour les entiers naturels a un plus petit élément $\perp$ (on ne sait rien sur l'entier en question) et un élément pour chaque entier (qui sont tous plus grands que $\perp$, mais incomparables entre eux) qui représentent le fait que l'on connaît la valeur de l'entier :



Un morphisme entre domaines de Scott est en gros une fonction croissante. En termes de ce que cela représente pour les programmes, cela signifie que, plus un programme a d'informations sur ses entrées, plus il peut produire d'information sur sa sortie. Par exemple, la *dénotation* (l'interprétation) d'un programme qui calcule le prédécesseur, mais boucle sur 0, serait la fonction qui envoie $\perp$ et 0 sur $\perp$ et chaque $n > 0$ sur $n - 1$.

Pour qu'une sémantique dénotationnelle soit considérée comme « bonne », elle doit satisfaire plusieurs critères. Supposons qu'on nous donne une relation sur les termes d'un langage qui dit si deux termes ont le même comportement (par exemple, ils renvoient toujours le même résultat quand on leur donne les mêmes entrées). La première propriété (et la plus évidente) qu'une sémantique dénotationnelle doit satisfaire est la *correction*, c'est-à-dire que deux programmes dont le comportement est différent doivent avoir des dénotations différentes. Il est très difficile de dire quoi que ce soit sur une sémantique qui ne respecte même pas cette propriété. La seconde propriété est la *complétude*, c'est-à-dire que deux programmes qui ont le même comportement doivent avoir des dénotations égales. Quand une sémantique satisfait ces deux propriétés, on dit qu'elle est *pleinement abstraite*. La pleine abstraction est une propriété intéressante dans le sens où il suffit d'étudier l'égalité dans le modèle pour en déduire si deux programmes ont des comportements équivalents. Une autre propriété intéressante est la *définissabilité* (compacte) du modèle, c'est-à-dire de savoir si tous les éléments du modèle sont l'image d'un terme du langage.

Enfin, une propriété cruciale de la sémantique dénotationnelle est la *compositionalité*, c'est-à-dire que la dénotation d'un programme peut se déduire de celles de ses sous-programmes. Par exemple, dans le cas du λ-calcul, on voudrait pouvoir calculer $[\![MN]\!]$ à partir de $[\![M]\!]$ et $[\![N]\!]$. Si l'on considère M comme une fonction de type $A \to B$ dont l'argument N est de type A, on est tenté de définir $[\![MN]\!]$ comme $[\![M]\!]\,([\![N]\!])$, ce qui est en effet une définition compositionnelle. (La véritable définition est en fait légèrement plus complexe car il faut interpréter des dérivations de typage, et non des termes.)

La sémantique aujourd'hui

Aujourd'hui, il existe un spectre impressionnant de modèles de langages de programmation, dont certains peuvent aussi bien être vus comme dénotationnels que comme opérationnels. L'exemple parfait de ce cas de figure est la sémantique de jeux : elle peut à la fois être vue comme une sémantique dénotationnelle parce qu'elle interprète les programmes par des *stratégies* sur une une notion générale de jeu, mais, les stratégies encodent aussi les possibles exécutions du programme, ce qui rend cette interprétation dynamique, et, dans de nombreux cas, les stratégies sont en bijection avec les formes normales du langage, ce qui rend le modèle très proche de la syntaxe, ce qui sont deux raisons de penser qu'elle est aussi proche de la sémantique opérationnelle.

Avec le temps, la définition des mots « dénotationnel » et « opérationnel » a changé pour prendre un sens plus large. Par exemple, il a longtemps été admis qu'un modèle qui n'est pas pleinement abstrait n'est pas un modèle dénotationnel (plus particulièrement s'ils ne sont pas complets, car si un modèle n'est pas correct, il ne mérite peut-être même pas le nom de modèle), mais beaucoup aujourd'hui considèrent ces modèles comme dénotationnels, dans une certaine mesure. Certains modèles sont indiscutablement dénotationnels (par exemple les domaines de Scott) alors que d'autres ne le sont pas du tout (par exemple les LTS). Entre ces deux extrêmes, il existe tout un continuum de modèles que l'on peut voir comme plus ou moins dénotationnels, et ce en s'appuyant principalement sur deux points. Le premier est à quel point les structures utilisées pour interpréter les types et les programmes est « mathématique ». Plus les modèles s'appuient sur des structures communes (par exemple les espaces topologiques ou les espaces vectoriels), plus ils sont considérés comme dénotationnels. Plus ils s'appuient sur des structures ad hoc (par exemple des LTS ou des catégories dérivées de la syntaxe du langage), moins ils sont considérés comme dénotationnels. Le second critère que l'on prend en compte est si le modèle a de bonnes propriétés (telles que la pleine fidélité) ou non : plus ils ont de bonnes propriétés, plus ils sont considérés dénotationnels. Sur ces deux points, la sémantique de jeux est dans une situation intermédiaire : les stratégies ne sont pas aussi communes que les espaces vectoriels ou topologiques, mais elles ne sont pas non plus des structures ad hoc, et bien que le modèle ne soit pas pleinement abstrait, il est compositionnel, indépendant de la syntaxe et un quotient extensionnel donne un modèle pleinement abstrait.

De façon similaire, la notion de modèle opérationnel a évoluée avec le temps. Auparavant, on considérait qu'un modèle était opérationnel si la structure pour interpréter les termes était dérivée de la syntaxe, comme c'est le cas pour les LTS. Aujourd'hui, les modèles opérationnels possèdent une dimension supplémentaire : un modèle est considéré comme opérationnel s'il est *dynamique*, c'est-à-dire que l'on peut retrouver l'exécution d'un programme à partir de son interprétation.

Il est probable que, de nos jours, la dichotomie entre modèles opérationnels et dénotationnels soit trop grossière pour être vraiment pertinente. Peut-être qu'une façon pertinente de la raffiner serait de définir plusieurs axes sur lesquels placer un modèle : selon qu'ils s'appuient sur des structures mathématiques communes (comme les espaces topologiques) ou soient syntaxiques (comme les LTS), qu'ils soient statiques (fonctions entre espaces) ou dynamiques (stratégies), qu'ils soient intentionnels (où l'égalité se dérive de la réduction des termes) ou extensionnels (où l'égalité se dérive d'une notion d'observation), etc.

Sémantique de jeux

Nous avons expliqué ce que sont la sémantique opérationnelle et la sémantique dénotationnelle (tout en restant à un niveau plutôt informel) et avons affirmé que la sémantique de jeux peut être vue comme les l'une ou l'autre. Nous allons maintenant expliquer avec un peu plus de détails ce qu'est la sémantique de jeux, car elle est au cœur de ce travail.

La naissance de la sémantique de jeux

La sémantique de jeux est née dans le monde de la logique, sous la forme de la *logique de dialogue* [75]. La logique de dialogue définit les preuves d'une formule comme deux entités qui débattent pour savoir si la formule est valide : *Joueur*, qui essaie de montrer que la formule est vraie, et *Opposant*, qui essaie de montrer qu'elle est fausse. Il s'agit d'un jeu formel qui décrit un débat entre deux personnes (sensées) qui débattent pour savoir si une proposition mathématique est vraie ou non : ils continuent de défendre leur position respective jusqu'à ce qu'un d'entre eux soit convaincu qu'il avait tort. Une formule est vraie si Joueur a une *stratégie gagnante* dans un certain jeu formel sur cette formule, ce qui revient à dire qu'il est toujours capable de convaincre Opposant que la formule est vraie, peu importe les objections levées par Opposant.

La logique de dialogues a ensuite été introduite dans le monde des langages de programmation sous le nom de *sémantique de jeux* par une longue liste d'auteurs, en particulier Berry et Curien [13] (sous le nom d'*algorithmes séquentiels*), qui sont les premiers à avoir utiliser la notion d'interaction en sémantique des langages de programmation et à donner une sémantique séquentielle et dénotationnelle d'un langage d'ordre supérieur, Blass [14, 15], qui a exhibé un lien entre sémantique de jeux et logique linéaire [41], Joyal [58], qui fut le premier a construire une catégorie de jeux et stratégies, Coquand [23], qui a lié la sémantique des jeux à la dynamique de l'élimination des coupures (et donc à l'évaluation des programmes), Abramsky, Jagadeesan et Malacaria [6], ainsi que

Hyland et Ong [56] et Nickau [87], qui ont construit les modèles de jeux les plus connus, et qui sont encore utilisés aujourd'hui : les jeux AJM et les jeux HON.

Les idées de base de la sémantique des jeux viennent directement de la logique de dialogues : les types sont interprétés par des jeux formels (que l'on peut voir comme des formules) et les programmes par des stratégies décrivant l'interaction du programme avec son environnement (et que l'on peut voir comme des preuves). Rentrons un peu plus dans les détails : les types sont interprétés par des *jeux* (que l'on appelle parfois des *arènes*) sur lesquels on définit une notion de *parties*. Les parties représentent toutes les interactions qu'un élément d'un type donné peut avoir avec son environnement. Les programmes sont ensuite interprétés par des *stratégies* dans ce jeu, c'est-à-dire que ce sont des ensembles de parties qui satisfont certaines contraintes. Ces parties sont toutes les interactions que ce programme peut avoir avec l'environnement. Ici, en revanche, à la différence de la logique de dialogues, les stratégies ne servent qu'à calculer des valeurs, et il n'y a pas de notion de stratégie « gagnante ».

Par exemple, on peut définir les parties sur le jeu des entiers naturels comme les suites $(q\ \mathbb{N})^*$, où q est un *coup* dans le jeu qui représente l'environnement qui demande la valeur du nombre (q pour « question ») et $\mathbb{N}$ est n'importe quel entier naturel, qui représente le programme retournant ce nombre comme résultat. La stratégie qui correspond à un compteur qui augmente de 1 chaque fois qu'il est appelé serait l'ensemble des parties de la forme $q\ 1\ q\ 2 \ldots q\ n$. Pour les fonctions de type `int → int`, on peut définir les parties comme les suites de la forme $(q_r\ (q_l\ \mathbb{N}_l)^*\ \mathbb{N}_r)^*$, où q_r est un coup qui représente l'environnement demandant le résultat de la fonction (r pour « *right* », comme dans la copie droite de `int`), $\mathbb{N}_r$ représente le programme renvoyant son résultat à l'environnement, q_l représente le programme demandant la valeur de son argument (l pour « left ») et $\mathbb{N}_l$ représente l'environnement donnant la valeur de l'argument à la fonction. Une séquence telle que décrite ci-dessus correspond donc à l'environnement demandant le résultat d'une fonction un certain nombre de fois, et, à chaque fois, la fonction demande à son tour la valeur de son argument un certain nombre de fois avant de retourner son résultat. Par exemple, la stratégie associée à la fonction successeur contiendrait toutes les parties de la forme $q_r\ q_l\ n_l^1\ (n^1+1)_r \ldots q_r\ q_l\ n_l^k\ (n^k+1)_r$. La structure exacte des parties dépend de quelle sémantique de jeux on utilise et importe peu ici, car les exemples ci-dessus donnent une bonne idée de ce que peuvent être des parties et des stratégies.

Deux autres idées sont présentes dans presque tous les modèles de jeux. La première est l'*innocence*, qui dit que les programmes purs (ceux qui n'utilisent que de la programmation purement fonctionnelle) sont interprétés par des stratégies *innocentes*. Le comportement de ces stratégies dépend d'une quantité d'information limitée sur l'histoire de la partie. L'information à laquelle une telle stratégie a accès encode le morceau de l'interaction entre le programme et l'environnement qui a mené à la situation présente. En particulier, une fonction ne ajuster son comportement qu'en fonction de ce qui s'est passé lors de l'appel courant à la fonction. Par exemple, la stratégie associée au programme du compteur décrit plus haut n'est pas innocente parce qu'elle doit se souvenir de la dernière réponse qu'elle a donné, alors qu'une stratégie innocente ne peut s'appuyer que sur q_r (et, en effet, le compteur est un programme impur). En revanche, la stratégie associée à la fonction successeur est innocente car sa réponse $(n^k + 1)_r$ ne dépend que de n_l^k.

Enfin, un aspect important de la sémantique de jeux est le façon de composer les stratégies. En effet, les modèles de jeux sont compositionnels, et il y a en particulier une notion de composition de stratégies qui correspond à la composition des programmes. Elle est définie en deux étapes appelées *composition parallèle* et *masquage*. La composition parallèle est ce qui permet aux stratégies d'interagir. Pour la définir, il faut une notion de « partie » sur trois jeux : si on se donne une stratégie σ sur les jeux A et B et une stratégie τ sur les jeux B et C, la composition parallèle $\sigma \| \tau$ est une stratégie sur les jeux A, B et C. Elle accepte de jouer une partie sur (A, B, C) si et seulement si σ accepte de jouer sa projection sur (A, B) et τ accepte de jouer sa projection sur (B, C). L'idée est que σ et τ communiquent sur le jeu B. La deuxième étape (le masquage), consiste à effacer ce qui se passe sur B pour faire une stratégies seulement sur les jeux A et C.

Comme nous l'avons déjà mentionné, les modèles de jeux peuvent à la fois être considérés comme dénotationnels parce que les programmes sont interprétés par des stratégies, qui sont des structures générales. D'un autre côté, ils sont aussi opérationnels parce que les stratégies sont souvent en bijection avec les formes normales et que l'interprétation des programmes correspond exactement à ses interactions avec l'environnement, ce qui en fait des modèles proches de l'exécution dynamique.

Le monde hétéroclite des modèles de jeux

Aujourd'hui, les modèles de jeux existants prennent racines dans des idées et des formalismes différents, ce qui donne un paysage très divers. Blass [14, 15] a construit ce qui est probablement considéré comme un des premiers modèles de jeux. Dans son approche, un jeu est décrit par l'arbre de ses positions et les parties sont les branches de cet arbre. Dans le cas des fonctions de type $A \to B$, les parties peuvent être vues comme la paire d'une branche dans A et d'une branche dans B. Une stratégie est juste le choix d'un coup à jouer à chaque position de l'arbre. La composition des stratégies est définie en jouant sur trois arbres en même temps (puis en masquant ce qui se passe sur l'arbre central). Malheureusement, la composition des stratégies n'est pas associative (à cause d'un problème technique sur la polarité des coups).

Les premiers modèles de jeux qui ont été largement utilisés, et dont des variantes sont encore utilisées de nos jours, sont les jeux AJM [6] et HON [56], qui ont tous deux été développés dans les années 90. Chez AJM, les jeux sont définis comme des ensembles de coups munis de conditions pour savoir quelles suites de coups sont des parties valides. Chez HON , les jeux sont définis comme des ensembles de coups structurés et les parties comme des séquences structurées (la structure exacte importe peu ici). Les deux modèles définissent ensuite les stratégies comme des ensembles de partis clos par préfixes et la composition de stratégies par composition parallèle et masquage (les deux opérations sont définies de manière similaire dans les deux modèles).

Aujourd'hui, il existe de nombreuses variantes des jeux HON et AJM, construites pour résoudre différents problèmes. Premièrement, un certain nombre de ces variantes, qui imposent des conditions supplémentaires pour qu'une séquence soit une partie valide, sont décrites dans la thèse de Harmer [46]. Certaines variantes ne sont pas couvertes par cette référence, par exemple [52], dans laquelle la polarité des coups est renversée. Certaines variantes enrichissent les jeux avec de la structure supplémentaire, par exemple des « liens copycat » [70], des actions de groupe [79, 7, 86] ou de la « cohérence » [68]. Certains modèles s'appuient sur les mêmes idées que les jeux AJM et HON, mais on peut insérer des jeux dans d'autres jeux, par exemple dans les *jeux polymorphes* [53], les *jeux variables* [4], les *jeux ouverts* [22] ou les *jeux contextuels* [71]. D'autres variantes ne peuvent être classées ni comme des jeux AJM, ni comme des jeux HON, mais s'appuient sur les mêmes idées, comme par exemple la catégorie séquoïdale construite dans [66]. Certains modèles définissent les jeux comme des arbres (un peu à la manière de Blass) et les stratégies comme des morphismes de jeux [54, 47].

De plus, la sémantique de jeux a donné naissance à de nombreux modèles concurrents. Cela parait naturel, dans le sens où un point fondamental de la concurrence est l'interaction des agents, et l'interaction est précisément au cœur de la sémantique de jeux. Certains de ces modèles s'appuient sur les jeux HON [67, 69, 40, 97], d'autres sur des structures plus exotiques. Parmi ces structures, on trouve les *structures d'événements* [88], dont un des points centraux est la notion de conflit entre événements, qui n'existe pas dans les modèles évoqués ci-dessus. Elles sont donc adaptées à l'étude de la concurrence et ont donné naissance à de nombreux modèles [92, 20, 21] intéressants. Grossièrement, elles décrivent les positions par des ensembles d'événements compatibles, les coups consistent à ajouter un événement à une position, et les stratégies sont des morphismes entre jeux. Melliès a fait un travail important pour comprendre le cœur de la sémantique des jeux avec ses *jeux asynchrones* [80, 82, 81, 84], qui s'appuient aussi sur des structures d'événements.

Les *terrains de jeux* [50, 29, 30] sont une autre structure exotique utilisée en sémantique de jeux. Les travaux rédigés dans cette thèse s'inspirent profondément de cette structure. Pour simplifier, les terrains de jeux sont des catégories doubles (en gros des catégories avec des morphismes verticaux et horizontaux et où la composition n'est définie qu'entre morphismes de la même classe) dont les objets sont les positions d'un jeu, les morphismes horizontaux sont les inclusions de positions et les morphismes verticaux les parties dans le jeu. Elles doivent satisfaire un certain nombre de propriétés pour que la construction fasse sens en tant que modèle de jeux. À partir d'un terrain de jeux, on peut définir de façon abstraite la catégorie $\mathbb{E}(X)$ des parties qui commencent à la position X. Les stratégies sont ensuite définies comme des préfaisceaux sur $\mathbb{E}(X)$, l'idée étant que cette définition est une généralisation de la notion standard de stratégie en tant qu'ensemble de parties clos par préfixe. Cette méthode a cependant un défaut qui est que, à ce jour, on ne sait pas comment composer les stratégies par composition parallèle et masquage. Tous les exemples connus de terrains de jeux font partie du cadre plus général dans lequel les parties sont définies comme des *diagrammes de cordes* (qui sont des formalisations de dessins intuitifs utilisés en sémantique des jeux). Dans ce cadre, les coups sont définis comme des diagrammes de corde de base et les parties comme des coups collés les uns aux autres. La plupart des travaux décrits dans cette thèse s'inscrivent dans cette lignée des diagrammes de cordes.

Motivation et contributions

L'idée principale qui a motivé ce travail a été de mieux comprendre le paysage des modèles de jeux. Plus précisément, alors qu'il y a de nombreux modèles de jeux, dont certains sont très similaires et d'autres basés sur des idées totalement différentes, il y a très peu de littérature qui tisse des liens entre ces différents modèles. Notre but a été de comprendre les modèles de jeux, en particulier à travers l'abstraction : nous avons essayé d'isoler des propriétés vérifiées dans différents modèles de jeux et d'en donner des versions abstraites pour étudier la classe des modèles qui vérifient ces propriétés pour prouver de façon abstraite des propriétés de tous ces modèles. Un autre point caractéristique de notre approche est notre utilisation constante d'outils catégoriques avancés pour rendre les constructions et les preuves plus efficaces. Pour cela, nous avons bénéficié de travaux précédents effectués dans le même état d'esprit, à savoir les récents travaux qui ont interprété l'innocence comme une condition de faisceau, grâce à une notion de morphisme entre parties due à Melliès. Un outil que nous introduisons pour la première fois en sémantique des jeux est la théorie des *carrés exacts* [45], qui est un outil clé dans notre travail et donne des preuves efficaces dans les chapitres 5 et 6.

Contributions principales

Nous donnons ici une liste des résultats principaux de chacune des contributions qui constituent cette thèse. Un peu plus bas, des sections sont dédiées à chaque contribution pour l'expliquer plus en détail.

Modèles fibrés Notre première contribution, étudiée dans le chapitre 4, suit le motif d'abstraction évoqué plus haut. Toutes les instances connues de modèles à base de diagrammes de cordes (un pour CCS [50], un pour le π-calcul [29], un pour les jeux HON, que nous étudions au chapitre 5 et un pour le join-calcul [36] qui reste non publié) sont construits de la même manière : on définit d'abord les positions comme des sortes de graphes, puis les coups comme des arêtes de dimension supérieure et enfin les parties comme des composées de coups. Nous appelons ces positions et parties des *diagrammes de cordes* car ils formalisent ce que les physiciens appellent ainsi. Pour pouvoir définir une catégorie de parties qui commencent à une position donnée, ces modèles doivent être *fibrés*, ce qui dit en gros que toute partie peut être restreinte aux sous-positions de sa position initiale. On donne d'abord une façon abstraite de construire des modèles à base de diagrammes de cordes à partir d'une description opérationnelle d'un langage qui généralise les constructions déjà existantes. On donne ensuite un critère nécessaire et suffisant pour que notre modèle soit fibré et un autre critère, seulement suffisant, mais plus simple à prouver. Cette contribution est basée sur [25].

Un pont entre modèles Notre seconde contribution principale, étudiée dans le chapitre 5, est une connexion entre deux variantes des jeux HON. La première variante [97] s'appuie sur la définition de *séquence justifiée*, qui est standard en sémantique des jeux, alors que le second suit l'approche à base de diagrammes de cordes telle que décrite dans notre contribution précédente. Il y a un lien informel évident entre les deux modèles, dans le sens où ils définissent tous deux les stratégies innocentes comme des faisceaux pour une topologie de Grothendieck induite par le plongement des vues dans les parties. On montre que ce lien peut être rendu plus solide : les deux modèles sont équivalents dans le sens où ils produisent des catégories de stratégies innocentes équivalentes. On donne d'abord un premier argument à base d'arbres de dérivation dans un calcul des séquents ad hoc qui décrit les jeux HON, mais l'argument est trop peu formel, et on donne donc ensuite une preuve plus formelle, sans utiliser les arbres de dérivation. Cette contribution se base sur [26] (qui donne l'argument avec les arbres de dérivation) et sa version longue [25] (qui donne la preuve directe).

Un noyau des modèles de jeux Notre dernière contribution principale, étudiée dans le chapitre 6, développe un cadre général pour étudier les modèles de jeux. L'idée est d'essayer de réduire les modèles de jeux à quelques propriétés basiques sur les parties et de retrouver les constructions principales des modèles de jeux à partir de ces propriétés de base. En se donnant une structure qui représente les parties, on montre comment définir un notion de stratégie concurrent et comment les composer en faisant une composition parallèle et un masquage. On montre que, sous certaines conditions, la composition des stratégies est associative et unitaire. On montre ensuite comment définir une catégorie de stratégies innocentes, sous réserve d'avoir un peu plus de structure. On montre aussi que ce beaucoup de modèles de jeux existants rentrent dans ce cadre. Cette contribution est basée sur [28].

Contributions qui n'apparaissent pas dans cette thèse

Nous donnons ici quelques contributions que l'on considère comme mineures ou qui ne sont pas étudiées dans cette thèse.

Diagrammes de cordes pour les traces concurrentes et les dépliages La première de ces contributions est décrite dans le chapitre 3, qui se base sur [24], et que l'on utilise comme une introduction aux diagrammes de cordes, qui seront très utilisés dans les chapitres 4 et 5. On montre comment les utiliser pour modéliser les traces concurrentes simplement, on donne quelques exemples d'applications de cette construction, et on l'illustre sur l'exemple des réseaux de Petri.

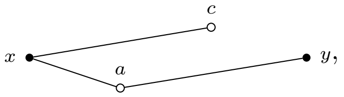
Un modèle de jeux pour le π-calcul Dans [29] et sa version longue [30], on construit un modèle du π-calcul qui est intentionnellement pleinement abstrait pour l'équivalence de tests équitable (la pleine abstraction intensionnelle est simplement la définissabilité, avec le sous-entendu qu'un quotient du modèle donne un modèle pleinement abstrait, quotient qui est par ailleurs souvent inévitable, vu que les modèles pleinement abstraits ne sont parfois pas récursivement présentables [74]). On définit d'abord une notion de partie pour le π-calcul à base de diagrammes de cordes (ce sera d'ailleurs l'exemple utilisé dans le chapitre 4), puis on montre comment interpréter les termes par des stratégies. Pour montrer que cette interprétation est intentionnellement pleinement abstraite, on utilise la théorie des *terrains de jeux*, qui nous donne de façon abstraite un LTS qui correspond à notre notion de stratégie, LTS que l'on relie à celui du π-calcul.

Plusieurs idées développées dans le chapitre 4 trouvent leur source dans ce papier, en particulier l'utilisation des systèmes de factorisation pour montrer que le modèle est fibré.

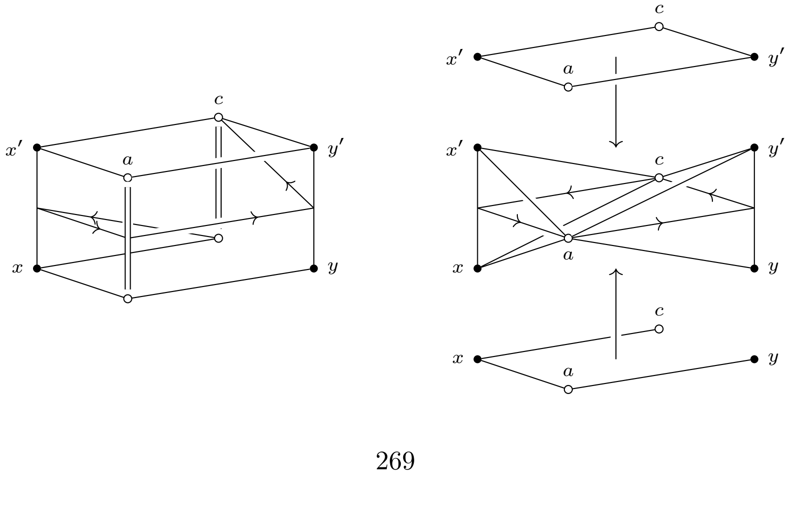
Interpréter les termes par des stratégies de façon abstraite Dans [27], on étudie l'interprétation des termes par des stratégies dans les modèles de jeux de façon abstraite. L'idée est de définir une notion de *paraterme* qui est suffisamment large pour inclure les vues, les parties et les termes. On définit ensuite l'interprétation des termes par des stratégies de façon abstraite sous la forme d'un *foncteur singulier*. Cette construction généralise des constructions existantes. On retrouve ensuite un résultat fondamental de la sémantique des jeux, la *définissabilité* (qui dit que toute stratégie innocente finie est isomorphe à l'interprétation d'une valeur), sous la forme de la *réalisation géométrique*.

Modèles fibrés

Dans le chapitre 4, on montre comment construire des modèles à base de diagrammes de cordes et montrer qu'ils sont fibrés sous certaines conditions. On suppose donnée une catégorie de base $\mathbb{C}$ qui décrit la sémantique opérationnelle d'un langage. Cette catégorie est équipée d'une notion de dimension. Les objets de petite dimension sont appelés les *joueurs* et les *canaux* et décrivent les *positions* du jeux, qui sont en gros des graphes de joueurs et de canaux. Les joueurs sont les agents de notre jeu et les canaux sont leurs moyens de communiquer entre eux. Par exemple, pour le π-calcul, une position est simplement une représentation de la topologie du réseau de communication entre les différents agents, comme dans :



qui représente un position avec deux joueurs x et y qui peuvent communiquer à travers le canal a, et x a connaissance d'un canal privé c. Les objets de dimension supérieure décrivent la dynamique du jeu. Par exemple, pour le π-calcul, une synchronisation dans laquelle x envoie le canal c sur a et où y le reçoit est dessinée comme à gauche ci-dessous.



Ici, la position initiale de la synchronisation est dessinée en bas et la position finale en haut, et l'on peut voir que, dans la position finale, l'avatar y' de y a connaissance du canal c que x a envoyé. La synchronisation est un des objets de dimension supérieure de la catégorie $\mathbb{C}$ correspondant au π-calcul. Pour chaque objet de dimension supérieure de $\mathbb{C}$, on se donne un tel dessin, que l'on appelle un *coup*. On appelle le foncteur qui associe à chaque objet de dimension supérieure un coup une *signature*. Une *partie* est une composée de coups (qui ne fait que coller plusieurs coups les uns aux autres). Formellement, les positions sont des préfaisceaux sur les deux premières dimensions de $\mathbb{C}$ (il s'agit bien d'une représentation formelle de la même chose car les dessins correspondent aux *catégories d'éléments* des objets de $\mathbb{C}$). Les coups sont des cospans $Y \to M \leftarrow X$ de préfaisceaux sur $\mathbb{C}$, où X est la position initiale, Y la position finale et M représente le coup. Par exemple, le cospan correspondant à la synchronisation dans le π-calcul est dessiné à droite de celle-ci, avec X en base, Y en haut, M au milieu et où tous les morphismes sont des inclusions.

À partir de n'importe quelle signature S, on construit une pseudo double catégorie $\mathbb{D}_{\mathsf{S}}$. Pour simplifier, il s'agit d'une structure mathématique qui possède un ensemble d'*objets* (ici, les positions), pour chaque paire d'objets X et Y, un ensemble de *morphismes horizontaux* $X \to Y$ (ici, les inclusions de la position X dans la position Y) et un ensemble de *morphismes verticaux* $Y \mathrel{-\!\!\bullet\!\!\to} X$ (ici, les parties dont la position initiale est X et la position finale Y). Elles ont aussi, pour tout carré tel que celui dessiné ci-dessous, un ensemble de *cellules* α (qui représentent ici le fait que u est incluse dans u' dans un certain sens).

$$\begin{tikzcd}
Y \arrow[r, "k"] \arrow[d, "u"'] & Y' \arrow[d, "u'"] \\
X \arrow[r, "h"'] & X'
\arrow[from=1-1, to=2-2, phantom, "\overset{\alpha}{\Longrightarrow}"]
\end{tikzcd}$$

On veut ensuite définir une catégorie $\mathbb{E}(X)$ des parties de position initiale X et dont les morphismes $u \to u'$ représenteraient le fait que u' est une extension de u (formellement, $\mathbb{E}(X)$ dépend de S, mais nous ne notons pas la dépendance pour ne pas alourdir la notation). La définition naturelle d'un morphisme de $u\colon Y \mathrel{-\!\!\bullet\!\!\to} X$ dans $u'\colon Y' \mathrel{-\!\!\bullet\!\!\to} X$ est une triplet d'un morphisme vertical $w\colon Z \to Y$, d'un morphisme horizontal $h\colon Z \to Y'$ et d'une cellule α comme ceci :

$$\begin{tikzcd}
Z \arrow[r, "h"] \arrow[d, "w"', "\bullet" marking] & Y' \arrow[dd, "u'", "\bullet" marking] \\
Y \arrow[d, "u"', "\bullet" marking] \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & \\
X \arrow[r, equal] & X.
\end{tikzcd}$$

Pour pouvoir définit une catégorie, il faut être capable de composer ces morphismes, ce qui nécessite de trouver une façon canonique de compléter le diagramme ci-dessous :

$$\begin{tikzcd}
Z'' \arrow[r, dashed] \arrow[d, dashed, "\bullet" marking] \arrow[dr, phantom, "\overset{\beta}{\Longrightarrow}"] & Z' \arrow[r, "h'"] \arrow[d, "\bullet" marking, "w'"] & Y'' \arrow[ddd, "\bullet" marking, "u''"] \\
Z \arrow[r, "h"] \arrow[d, "\bullet" marking, "w"'] & Y' \arrow[dd, "\bullet" marking, "u'"] \arrow[dr, phantom, "\overset{\alpha'}{\Longrightarrow}"] & \\
Y \arrow[d, "\bullet" marking, "u"'] \arrow[r, phantom, "\overset{\alpha}{\Longrightarrow}"] & & \\
X \arrow[r, equal] & X \arrow[r, equal] & X.
\end{tikzcd}$$

Il faut donc être capable de restreindre toute partie w' à une sous-position Z de sa position initiale. Plus précisément, on veut que, pour toute partie $u: Y \xrightarrow{\bullet} X$ et morphisme $X' \to X$, il existe une partie $u': Y' \xrightarrow{\bullet} X'$ et une cellule α comme ci-dessous telle que, pour tout diagramme de la forme

$$\begin{tikzcd}
Y'' \arrow[drr, "k'"] \arrow[dr, dashed, "k''"'] \arrow[dd, "\bullet" marking, "u''"'] & & \\
& Y' \arrow[r, "k"'] \arrow[dd, "\bullet" marking, "u'"'] \arrow[r, phantom, shift right=6, "\overset{\alpha'}{\Longrightarrow}"] \arrow[dr, phantom, "\overset{\alpha}{\Longrightarrow}"] & Y \arrow[dd, "\bullet" marking, "u"] \\
X'' \arrow[dr, "h''"'] \arrow[drr, "h'" near end] \arrow[ur, phantom, "\overset{\alpha''}{\Longrightarrow}"] & & \\
& X' \arrow[r, "h"'] & X,
\end{tikzcd}$$

il existe un morphisme pointillé et une cellule correspondante (ce qui signifie que u' est bien une restriction de u le long de h, ce qui est nécessaire pour que la composition soit bien définie dans $\mathbb{E}(X)$). On dit d'un modèle qui vérifie cette propriété qu'il est *fibré*.

Pour prouver cette propriété, on utilise les *systèmes de factorisation* [17]. Cet outil algébrique permet de factoriser tout morphisme d'une catégorie sous la forme $r \circ l$, où l et r appartiennent à des classes $\mathcal{L}$ et $\mathcal{R}$ fixées et qui sont *orthogonales*, ce qui signifie qu'elles possèdent une certaine propriété de relèvement. On construit un système de factorisation dont la classe $\mathcal{L}$ est générée par les jambes $X \to M$ de tous les cospans $Y \to M \leftarrow X$ de notre signature (c'est-à-dire tous les coups). Comme les parties $u: Y \xrightarrow{\bullet} X$ sont des cospans de préfaisceaux $Y \to U \leftarrow X$, on est en fait en face de la situation décrite par le diagramme à gauche ci-dessous, que l'on complète en factorisant $l \circ h$ en $h' \circ l'$ (à l'aide du système de factorisation) puis en prenant le produit fibré de f et h'.

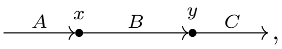
$$\begin{tikzcd}
Y' \arrow[r, dashed, "h''"'] \arrow[d, dashed, "f'"'] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d, "f"] \\
U' \arrow[r, dashed, "h'"] & U \\
X' \arrow[u, dashed, "l'"] \arrow[r, "h"] & X \arrow[u, "l"']
\end{tikzcd}
\qquad
\begin{tikzcd}
Y'' \arrow[drr, bend left, "q''"] \arrow[dr, dashed, "s''"'] \arrow[dd, "f''"'] & & \\
& Y' \arrow[r, "h''"'] \arrow[dd, "f'"' near start] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[dd, "f"] \\
U'' \arrow[drr, bend left, "q'"] \arrow[dr, dashed, "s'"'] & & \\
& U' \arrow[r, "h'"] & U \\
X'' \arrow[uu, "l''"] \arrow[drr, bend left, "q"] \arrow[dr, "s"'] & & \\
& X' \arrow[uu, "l'"] \arrow[r, "h"] & X \arrow[uu, "l"']
\end{tikzcd}$$

On obtient la propriété désirée en construisant s' et s'' dans le diagramme de droite : le premier vient de la propriété de relèvement de notre système de factorisation et le deuxième de la propriété universelle du produit fibré.

Il reste ensuite à prouver que $Y' \to U' \leftarrow X'$ est une partie, ce que l'on fait par récurrence, en supposant que c'est le cas pour les coups. On donne ensuite un critère suffisant sur $\mathbb{C}$ pour que la restriction soit une partie.

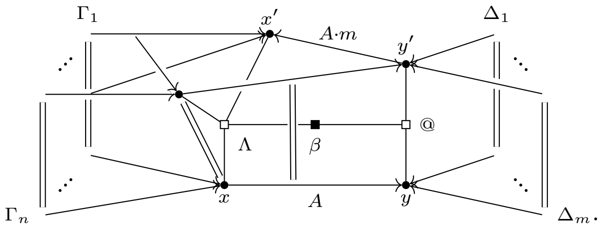
Un pont entre modèles

Dans le chapitre 5, on commence par construire une variante des jeux HON basée sur les diagrammes de cordes, tel qu'expliqué dans A, puis on construit une traduction entre cette variante et une autre (basée sur la notion standard de séquence justifiée), à la fois au niveau des parties et au niveau des stratégies. On part d'un calcul des séquents qui décrit les jeux d'arènes et qui rappelle fortement un calcul des séquents focalisé pour la logique intuitionniste. On dérive une signature S_{HON} de ce calcul des séquents. Ici, les canaux sont des jeux à deux joueurs, que l'on dessine comme des arêtes entre les deux joueurs, que l'on dessine comme des sommets. Les positions ressemblent typiquement à



où, dans cette position particulière, x joue comme Joueur sur B et Opposant sur A et y comme Joueur sur C et Opposant sur C. Les arêtes pendantes représentent l'interaction avec l'environnement. Cette position représente typiquement une fonction de type $A \to B$, modélisée par x, et une de type $B \to C$, modélisée par y. Les joueurs qui n'ont que des arêtes entrantes représentent les parties du programme qui sont en train de calculer, alors que celles qui possèdent une arête sortante attendent qu'une autre partie du programme l'appelle (sur cette arête sortante).

La dynamique de ce jeu est dérivée de l'élimination des coupures de notre calcul et est dessinée :



La façon la plus simple de comprendre cette interaction est probablement d'un point de vue de l'exécution des programmes. Dans la position initiale (en bas du dessin), x est une fonction de type $\Gamma_1 \to \ldots \to \Gamma_n \to A$ qui produit des résultats de type A en ayant accès à des ressources de type $\Gamma_1, \ldots, \Gamma_n$, alors que y est une partie de programme en train de calculer et a accès à des ressources de types $\Delta_1, \ldots, \Delta_m$ et A (produite par x). L'interaction représente y qui demande à x sa valeur de retour. Dans la position finale (en haut du dessin), les polarités des deux joueurs ont changé : l'avatar x' de x est maintenant en train de calculer, alors que l'avatar y' de y attend que x' ait fini de calculer et lui renvoie une valeur (c'est la raison pour laquelle x' a « accès » à y' : il s'agit d'un continuation).

Remarquons qu'y' a toujours accès à x, au cas où il aurait besoin de le rappeler plus tard lors de son exécution.

On dérive de cette signature une pseudo double catégorie $\mathbb{D}_{HON}$ dont les objets sont les positions, les morphismes horizontaux les inclusions de positions et les morphismes verticaux les parties, tel que décrit dans A. On montre ensuite que S_{HON} vérifie les conditions pour que $\mathbb{D}_{HON}$ soit fibrée, d'où on dérive une catégorie $\mathbb{E}(X)$ de parties de position initiale X. En particulier, on dérive des catégories $\mathbb{E}(A \vdash B)$ au-dessus de chaque position de la forme

$$\xrightarrow{A} \overset{x}{\bullet} \xrightarrow{B} .$$

On dérive aussi des sous-catégories $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ qui contiennent uniquement les *vues*, qui sont des partie particulières, définies de façon légèrement ad hoc. On définit ensuite les stratégies comme les préfaisceaux sur $\mathbb{E}(A \vdash B)$ (comme dans les autres modèles à base de diagrammes de cordes) et les stratégies innocentes comme les préfaisceaux qui sont dans l'image de $\prod_{\mathsf{i}} : \widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \to \widehat{\mathbb{E}(A \vdash B)}$, où i est le plongement de $\mathbb{E}^{\mathbb{V}}(A \vdash B)$ dans $\mathbb{E}(A \vdash B)$, où $\widehat{\mathbb{C}}$ est la catégorie des préfaisceaux sur $\mathbb{C}$ et $\prod$ est l'extension de Kan à droite.

Il existe aussi des catégories standard correspondant à celles-ci dans le cadre des jeux HON basés sur des séquences justifiées : la catégorie $\mathbb{P}_{A,B}$ des parties sur les arènes (A, B) et la catégorie $\mathbb{V}_{A,B}$ des vues. De façon similaire, les stratégies sont définies comme des préfaisceaux sur $\mathbb{P}_{A,B}$ et les stratégies innocentes comme celles qui sont dans l'image de $\prod_{\mathsf{i}_{HON}}$ (où i_{HON} est le plongement de $\mathbb{V}_{A,B}$ dans $\mathbb{P}_{A,B}$).

Durant la plus grande partie du reste du chapitre, on construit un carré commutatif comme à gauche ci-dessous, où F est un plongement plein et $F^{\mathbb{V}}$ est une équivalence de catégories.

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r, hook, "\mathsf{i}_{TO}"] \arrow[d, "F^{\mathbb{V}}"'] & \mathbb{P}_{A,B} \arrow[d, "F"] \\
\mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[r, hook, "\mathsf{i}"'] & \mathbb{E}(A \vdash B)
\end{tikzcd}
\qquad
\begin{tikzcd}
\widehat{\mathbb{V}_{A,B}} \arrow[r, hook, "\Pi_{\mathsf{i}_{TO}}"] & \widehat{\mathbb{P}_{A,B}} \\
\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)} \arrow[r, hook, "\Pi_{\mathsf{i}}"'] \arrow[u, "\Delta_{F^{\mathbb{V}}}"] & \widehat{\mathbb{E}(A \vdash B)} \arrow[u, "\Delta_F"']
\end{tikzcd}$$

En particulier, on a que $\widehat{\mathbb{E}^{\mathbb{V}}(A \vdash B)}$ et $\widehat{\mathbb{V}_{A,B}}$ sont équivalentes à travers le foncteur de restriction $\Delta_{F^{\mathbb{V}}}$. Mais on peut même aller plus loin : comme $F^{\mathbb{V}}$ est une équivalence de catégories et F est plein et fidèle, le carré est *exact* [45], ce qui veut dire que le carré de droite commute à isomorphisme près. Cela signifie que les catégories de stratégies innocentes des deux modèles sont équivalentes et que cette équivalence est compatible avec les foncteurs de *saturation* $\prod_{\mathsf{i}}$ et $\prod_{\mathsf{i}_{HON}}$. Les différences entre ces deux variantes sont donc principalement de l'ordre de la présentation.

La partie difficile de ce travail est de définir les foncteurs F et $F^{\mathbb{V}}$ évoqués ci-dessus. Le second est simplement défini comme la restriction du premier aux vues, donc la partie la plus difficile est de définir F. On donne deux façons de le faire : d'abord en utilisant un argument informel, puis en donnant une preuve formelle.

La première façon fait appel à des arbres de dérivation pour un calcul des séquents ad hoc. On définit une catégorie $\mathbb{T}(A \vdash B)$ dont les objets sont les arbres de conclusion $(A \vdash B)$ et dont les morphismes sont les inclusions d'arbres, et une sous-catégorie $\mathbb{B}(A \vdash B)$ de branches de conclusion $(A \vdash B)$. On décompose le carré en

$$\begin{tikzcd}
\mathbb{V}_{A,B} \arrow[r] \arrow[d, "\mathsf{i}_{TO}"'] & \mathbb{B}(A \vdash B) \arrow[r] \arrow[d] & \mathbb{E}^{\mathbb{V}}(A \vdash B) \arrow[d, "\mathsf{i}"] \\
\mathbb{P}_{A,B} \arrow[r] & \mathbb{T}(A \vdash B) \arrow[r] & \mathbb{E}(A \vdash B)
\end{tikzcd}$$

en montrant d'abord que $\mathbb{T}(A \vdash B)$ est équivalente à $\mathbb{E}(A \vdash B)$, que cette équivalence se restreint à une équivalence entre $\mathbb{B}(A \vdash B)$ et $\mathbb{E}^{\mathbb{V}}(A \vdash B)$, puis en construisant un plongement plein de $\mathbb{P}_{A,B}$ dans $\mathbb{T}(A \vdash B)$ et en prouvant qu'il se restreint à une équivalence entre $\mathbb{V}_{A,B}$ et $\mathbb{B}(A \vdash B)$. Cet argument n'est cependant pas totalement satisfaisant au sens où les arbres ne sont pas traités suffisamment formellement et un traitement formel ne rendrait pas le problème plus simple à résoudre que sans utiliser $\mathbb{T}(A \vdash B)$.

On donne donc une construction formelle de F et $F^{\mathbb{V}}$ sans passer par $\mathbb{T}(A \vdash B)$. Les preuves de ce chapitre sont beaucoup plus ad hoc que dans le reste de cette thèse, ce qui n'est pas très surprenant, dans le sens où on essaie de tisser du lien entre deux modèles qui sont construits en utilisant des méthodes très différentes.

Un noyau des modèles de jeux

Dans le chapitre 6, on décrit comment définir plusieurs catégories de stratégies à partir d'une description basique d'un modèle de jeux. On se donne un foncteur $\mathbb{P}$ qui décrit un modèle de jeux. Pour tout jeu A, $\mathbb{P}$ donne une catégorie $\mathbb{P}_A$ des parties sur le jeu A. Il donne aussi, pour tous jeux A et B, une catégorie $\mathbb{P}_{A,B}$ de parties sur la paire de jeux (A, B), et ainsi de suite pour les triplets et quadruplets de jeux. Il donne aussi des foncteurs d'insertion, par exemple $\iota_0 : \mathbb{P}_{A,B} \to \mathbb{P}_{A,A,B}$ qui, typiquement, ne fait que dupliquer ce qui se passe sur A ; et des foncteurs de suppression, par exemple $\delta_1 : \mathbb{P}_{A,B,C} \to \mathbb{P}_{A,C}$ qui, typiquement, efface ce qui se passe sur B.

On dérive de façon abstraite une notion de stratégie d'un tel $\mathbb{P}$: une stratégie sur la paire de jeux (A, B) est un préfaisceau sur $\mathbb{P}_{A,B}$ (on étudie les stratégies sur les paires de jeux pour montrer qu'elles forment une catégorie, mais on pourrait définir des stratégies sur un seul jeu de la même façon). L'idée est qu'une stratégie σ accepte une partie p si $\sigma(p) \neq \varnothing$ (plus précisément, $\sigma(p)$ est l'ensemble des états dans lesquels la stratégie peut se retrouver après avoir joué p). Pour montrer que les jeux et stratégies forment une catégorie, on définit la composition en tant que composition parallèle plus masquage et des identités pour cette composition, connues sous le nom de stratégies *copycat*.

La composition parallèle $\sigma \| \tau$ de deux stratégies σ sur (A, B) et τ sur (B, C) accepte de joueur une *séquence d'interaction* (une partie sur trois jeux) u si et seulement si σ accepte de jouer la projection $\delta_2(u)$ du u sur (A, B) et τ accepte de jouer la projection $\delta_0(u)$ du u sur (B, C). Le masquage d'une stratégie σ sur $\mathbb{P}_{A,B,C}$ accepte de jouer une partie p si et seulement s'il existe une séquence d'interaction u qui se projette sur p et que σ accepte de jouer, c'est-à-dire qu'elle accepte de jouer les mêmes parties que σ, sauf que l'on masque se qui se passe sur B. Enfin, la stratégie copycat sur le jeu A est la stratégie sur $\mathbb{P}_{A,A}$ qui « recopie » tous les coups joués par Opposant.

Dans le cadre que nous définissons dans ce chapitre, la composition et les stratégies copycat sont toutes deux définies par des *foncteurs polynomiaux.* Étant donné un foncteur $F:\mathbb{C} \to \mathbb{D}$, le foncteur de restriction $\Delta_F:\widehat{\mathbb{D}} \to \widehat{\mathbb{C}}$ est donné par pré-composition par F^{op}. Il admet des adjoints à gauche et à droite $\sum_F:\widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$ et $\prod_F:\widehat{\mathbb{C}} \to \widehat{\mathbb{D}}$, appelés *extensions de Kan à gauche* et *à droite*, respectivement. Pour comprendre $\sum_F$ à un niveau intuitif, il faut imaginer que le préfaisceau $\sum_F(X)$ est non-vide au-dessus d'un élément d s'il existe un antécédent c de d tel que X est non-vide au-dessus de c. Pour $\prod_F$, l'intuition est que X doit être non-vide au-dessus de tous les antécédents de d. On peut donc penser à ces foncteurs comme à des foncteurs $\exists$ et $\forall$. Un foncteur de $\widehat{\mathbb{C}}$ dans $\widehat{\mathbb{D}}$ est polynomial si c'est une composée d'un nombre quelconque de restrictions et extensions de Kan à gauche et à droite.

La composition doit être un foncteur de $\widehat{\mathbb{P}_{A,B}} \times \widehat{\mathbb{P}_{B,C}}$ dans $\widehat{\mathbb{P}_{A,C}}$. On peut la voir de façon équivalente comme un foncteur de $\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}}$ dans $\widehat{\mathbb{P}_{A,C}}$. On la définit comme la composée

$$\widehat{\mathbb{P}_{A,B} + \mathbb{P}_{B,C}} \xrightarrow{\Delta_{\delta_2+\delta_0}} \widehat{\mathbb{P}_{A,B,C} + \mathbb{P}_{A,B,C}} \xrightarrow{\Pi_\nabla} \widehat{\mathbb{P}_{A,B,C}} \xrightarrow{\Sigma_{\delta_1}} \widehat{\mathbb{P}_{A,C}},$$

où ∇ est le foncteur codiagonal. Il s'agit bien d'une définition polynomiale, et l'idée derrière cette définition est exactement celle de la composition parallèle suivie du masquage, ce que l'on peut voir en calculant de ce que fait ce foncteur avec la description des extensions de Kan qu'on a donnée plus haut. La composée des deux premiers foncteurs est la composition parallèle : si on appelle θ l'image de $[\sigma, \tau]$, alors θ accepte de jouer une séquence d'interaction u si et seulement si, pour chacun de ses antécédents (c'est-à-dire $\mathrm{inl}\, u$ et $\mathrm{inr}\, u$), $[\sigma, \tau]$ accepte de jouer $(\delta_2+\delta_0)(u')$, c'est-à-dire que σ accepte de jouer $\delta_2(u)$ et τ accepte de jouer $\delta_0(u)$. Le dernier foncteur $\sum_{\delta_1}$ est le masquage : si on appelle τ l'image de σ, alors τ accepte de jouer p si et seulement s'il existe une séquence d'interaction u qui se projette sur p et est acceptée par σ.

Les stratégies copycat sont aussi définies comme des foncteurs polynomiaux. On peut voir la stratégie copycat sur A comme un foncteur de 1 vers $\widehat{\mathbb{P}_{A,A}}$. Comme $1 \cong \widehat{\varnothing}$, on peut la définir comme :

$$\widehat{\varnothing} \xrightarrow{\Pi_!} \widehat{\mathbb{P}_A} \xrightarrow{\Sigma_{\iota_0}} \widehat{\mathbb{P}_{A,A}}.$$

L'idée est que $\prod_!$ est le préfaisceau terminal sur $\mathbb{P}_A$, donc il accepte toutes les parties sur $\mathbb{P}_A$, et $\sum_{\iota_0}(\sigma)$ accepte une partie p si et seulement si p est de la forme $\iota_0(p')$ et σ accepte p'. Comme ι_0 correspond typiquement à recopier ce qui se passe sur une copie de A sur l'autre, la composée est bien la stratégie copycat sur A : elle accepte de jouer p si et seulement si Joueur recopie tout ce qu'Opposant fait.

Notre but est ensuite de montrer que les jeux et les stratégies forment une catégorie dont la composition et les identités sont celles qu'on vient de définir, ce qui signifie que la composition doit être associative et neutre pour les stratégies copycat. On prouve que c'est le cas sous certaines conditions. La condition principale est inspirée de la méthode traditionnelle pour prouver que la composition des stratégies est associative, à savoir le lemme du *zipping*, qui dit que, dans certains cas, si deux séquences d'interaction se projettent sur la même partie, alors il existe une unique *séquence d'interaction généralisée* (une partie sur quatre jeux) qui se projette sur ces deux séquences d'interaction.

Nous faisons tout ceci pour notre notion de stratégie concurrente et nous voudrions obtenir les mêmes résultats pour les stratégies « traditionnelles », que l'on voit comme des foncteurs $\mathbb{P}_{A,B}^{op} \to 2$, où 2 est l'ordinale $0 \to 1$ vu comme une catégorie. On dérive du fait que les jeux et stratégies concurrentes forment une catégorie que c'est aussi le cas pour les jeux et stratégies traditionnelles. On étudie également un certain nombre de modèles de jeux pour montrer qu'ils rentrent dans ce cadre et que la composition des stratégies définie abstraitement dans notre cadre correspond à la composition dans ces différents modèles.

Enfin, on s'attaque à la question de l'innocence. On suppose donnée une sous-catégorie pleine $\mathbb{V}_{A,B} \xrightarrow{\mathsf{i}_{A,B}} \mathbb{P}_{A,B}$ de vues pour chaque catégorie $\mathbb{P}_{A,B}$. On définit ensuite les stratégies innocentes comme les préfaisceaux dans l'image de $\widehat{\mathbb{V}_{A,B}} \xrightarrow{\Pi_{\mathsf{i}_{A,B}}} \widehat{\mathbb{P}_{A,B}}$. L'idée derrière cette définition est qu'un préfaisceau $\prod_{\mathsf{i}_{A,B}}(\sigma)$ accepte de jouer p si et seulement si, pour tout morphisme $v \to p$ d'une vue v dans p, σ accepte de jouer v. Pour que cette définition soit la bonne, il faut que $\mathbb{P}_{A,B}$ contienne suffisamment de morphismes de parties (dans le cas des jeux HON, il s'agit des morphismes donnés par Melliès [80], ensuite réutilisés par Levy [73] puis Tsukada et Ong [96]). Un tel préfaisceau accepte de jouer une partie p si et seulement s'il accepte de jouer toutes les vues de p, ce qui est l'idée même de l'innocence. On souhaite montrer que les jeux et les stratégies innocentes forment une sous-catégorie de la catégorie des jeux et stratégies, ce que l'on fait en imposant que le modèle vérifie certaines propriétés.

Il est peut-être intéressant de voir quelles méthodes on utilise pour prouver ce genre de résultat. Prenons par exemple la préservation de l'innocence, qui dit que la composée de deux stratégies innocentes est encore innocente. Elle est prouvée en étudiant le diagramme suivant. (Il n'est pas nécessaire de comprendre ce diagramme.)

$$\begin{tikzcd}
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[rr, "\Pi"] \arrow[d, equal] & & \mathbb{V}_{(A,B),(B,C)} \arrow[d, "\Pi"'] & \mathbb{V}_{A,B,C} \arrow[l, "\Delta"'] \arrow[r, "\Sigma"] \arrow[d, "\Pi"'] & \mathbb{V}_{A,C} \arrow[d, "\Pi"] \\
\mathbb{V}_{A,B} + \mathbb{V}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{A,B} + \mathbb{P}_{B,C} \arrow[r, "\Pi"'] & \mathbb{P}_{(A,B),(B,C)} & \mathbb{P}_{A,B,C} \arrow[l, "\Delta"] \arrow[r, "\Sigma"'] & \mathbb{P}_{A,C}
\end{tikzcd}$$

Remarquons d'abord que la question de savoir si le diagramme commute ou non n'a aucun sens parce qu'il n'y a ni point de départ, ni point d'arrivée dans ce diagramme. En revanche, quand on passe aux catégories de préfaisceaux (où les Δ, $\sum$ et $\prod$ servent à dire quoi faire sur les foncteurs), les flèches étiquetées Δ se retournent, et on peut se demander si le diagramme commute. Sur les catégories de préfaisceaux, la ligne du bas correspond à prendre deux stratégies innocentes et les composer, donc, si le diagramme commute, la composée de deux stratégies innocentes est dans l'image de $\prod_{\mathsf{i}_{A,C}}$, et donc innocente. Dans le diagramme entre catégories de préfaisceaux, le carré de gauche commute parce que la carré sous-jacent commute et celui du milieu parce que le carré sous-jacent est exact. Il ne nous reste donc qu'à montrer que le carré de droite commute, ce qui est plus difficile. Quasiment toutes les preuves de ce chapitre suivent le même schéma. On étudie le diagramme sous-jacent et

- pour les carrés faits seulement de Δ (ou de $\prod$ ou de $\sum$), on montre que le diagramme sous-jacent commute,

- pour les carrés de la forme

$$\begin{tikzcd}
A \arrow[r, "\Pi"] & B \\
C \arrow[u, "\Delta"] \arrow[r, "\Pi"'] & D \arrow[u, "\Delta"']
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
A \arrow[r, "\Sigma"] & B \\
C \arrow[u, "\Delta"] \arrow[r, "\Sigma"'] & D \arrow[u, "\Delta"']
\end{tikzcd}$$

  on montre que le carré est exact, ce qui montre que le carré entre catégories de préfaisceaux commute à isomorphisme près,

- pour les carrés de la forme

$$\begin{tikzcd}
A \arrow[r, "\Sigma"] \arrow[d, "\Pi"'] & B \arrow[d, "\Pi"] \\
C \arrow[r, "\Sigma"'] & D
\end{tikzcd}$$

  on doit utiliser des preuves plus complexes.

Abstract

Game semantics is a class of models of programming languages in which types are interpreted as games and programs as strategies. Such game models have successfully covered diverse features, such as functional and imperative programming, or control operators. They have recently been extended to non-deterministic and concurrent languages, which generated an in-depth recasting of the standard approach: plays are now organised into a category, on which strategies are presheaves. The fundamental notion of innocence has also been recast as a sheaf condition. This thesis is a study of various constructions appearing in this new approach to game semantics.

We first consider a pattern common to several game models of concurrent languages, in which games and plays are first organised into a double category, from which strategies are then derived. We provide an abstract construction of such a double category from more basic data, and prove that, under suitable hypotheses, the result allows the construction of strategies.

Our second contribution is to relate two established techniques for defining plays: the standard one, based on justified sequences, and a more recent one, based on string diagrams. We (fully) embed the former into the latter and prove that they induce essentially the same model.

Finally, we propose an axiomatisation of the notions of game and play, from which we formally derive a category of games and strategies. We also refine the axioms to deal with innocence, and prove that, under suitable hypotheses, innocent strategies are stable under composition.

Résumé

La sémantique des jeux est une approche pour modéliser les langages de programmation dans laquelle les types sont interprétés par des jeux et les programmes par des stratégies. Ces modèles de jeux ont couvert des constructions fonctionnelles et impératives, des opérateurs de contrôle, etc. L'approche a récemment été étendue aux langages non-déterministes et concurrents, provoquant au passage un changement de perspective profond : les parties sont maintenant organisées en une catégorie, sur laquelle les stratégies sont des préfaisceaux. La notion fondamentale d'innocence a aussi été caractérisée comme une condition de faisceau. Cette thèse s'attache à l'étude de quelques constructions apparaissant dans ces nouveaux modèles de jeux.

D'abord, constatant que, dans plusieurs de ces modèles, l'étape cruciale consiste à définir une catégorie double de jeux et de parties, nous proposons une construction abstraite d'une telle catégorie double à partir de données de base, puis nous démontrons que, sous des hypothèses adéquates, le résultat obtenu permet en effet la construction des stratégies.

Dans un second temps, nous établissons un lien entre deux techniques existantes pour définir les parties : la technique standard, fondée sur les séquences justifiées, et une autre plus récente utilisant les diagrammes de cordes. Nous définissons un plongement (plein) de la première dans la seconde et prouvons qu'elles induisent essentiellement le même modèle.

Enfin, nous proposons une axiomatisation des notions de jeu et de partie, de laquelle nous tirons une catégorie de jeux et stratégies. Nous raffinons ensuite les axiomes pour traiter l'innocence et nous démontrons que, sous des hypothèses adéquates, les stratégies innocentes sont stables par composition.