Concurrent structures in game semantics

Simon Castellan

Contents

CHAPTER 1

Introduction

"Shall we go to the pub next Wednesday?" is a dreadful invitation to hear on a Monday. Are we going to the pub in two days, or in nine? Different people mean different things; yet some feel very strongly about the lack of ambiguity of this formulation. Marking both days on one's agenda is often the only way to save oneself from the social *faux-pas* of asking for clarifications.

This illustrates the gap between what is *meant* and what is *said*. To ease communication, languages often contain mechanisms of *indirections*, that make decoding the meaning of a sentence sometimes very difficult. *Next Wednesday* is such an indirection because it cannot be decoded without contextual information about today. One important function of these indirections is to save time, to avoid repeating something that the interlocutor already knows.

Those misunderstandings exist in spite of our natural languages being very close to our thought process. Imagine now that, instead of communicating with one of your peers, aware of these social conventions, you have to communicate a recipe to someone simple-minded, who takes the words "pot", "pan", and any other cooking-related word literally, without question. To make sure your message gets across requires great care, as one is used to employ shortcuts that will not be correctly understood. "Boil water in a pan" might be understood as "boil water in a frying pan" which would not be the intended meaning.

This is what programming feels like. Programming is turning one's thought into a series of basic instructions hopefully achieving the desired outcome. This conceptual gap, between the abstract thought and the concrete program is very hard to jump, and any mistake can lead to bugs having serious consequences.

A sentence (or a program) is in the end simply a sequence of arbitrary characters obeying a set of rules, called the **syntax** of the language. The meaning of such a sequence of characters (forming a syntactically-valid sentence) is called its **semantics**: it is what the sentence means (to us), whereas syntax is the means by which it is communicated. Extracting the semantics of a sentence is a difficult problem: even humans sometimes struggle as pointed out in the first paragraph.

However, due to their formal nature, it is somewhat simpler to extract the meaning of a program, explaining the behaviour of the program. Some programming languages even explain how to **define** *precisely* the meaning of programs written in that language. This allows a programmer to reason accurately about their program as they can check that it coincides with what they had in mind.

Most programming languages however, do not offer such means of reasoning about the behaviour of the program, making it hard to be sure it is correct. Mainstream programming languages support a combination of features such as control, higher-order, concurrency (with shared memory) that makes it hard to define **precisely** the behaviour of programs in general. Indeed, this combination

of features often result in complex behaviours hard to analyze. Explaining how to define the semantics of programs in general, proved to be a mathematically challenging task for programming language designers. They struggle to explain exhaustively how programs of their language should behave, making it hard to be confident a given program is correct. Being able to define precisely the semantics of all valid programs of a language is also extremely important to implement meaning-preserving translators between programming languages, or compilers.

In this thesis, we propose new mathematical tools, that help giving a semantics to complex programming languages, in a manner that is **compositional**: the meaning of a program should depend only on the meaning of its subparts. This requirement ensures key properties, relevant both in practice and in theory.

In the rest of this introduction, we introduce the technical tool used – *partial-order based game semantics*, on the simple example of arithmetic expressions.

Arithmetic expressions and operational semantics. An arithmetic expression is a well-bracketed *unevaluated* sequence of numbers and mathematical operators. For instance "$\underline{2}$+($\underline{1}$+$\underline{1}$)" is an arithmetic expression, distinct from the expression "$\underline{4}$", even though they both evaluate to the same number, 4. Underlining is used to distinguish a number (the *mathematical* object) from the corresponding expression (the *syntactic* object). This distinction also arises in natural language: "four" is the syntactic representation of the number 4.

When we write down an arithmetic expression, what we mean is a number. I can write "$\underline{365}\times\underline{24}$" to mean the number of hours in a year, instead of "$\underline{8760}$" because I prefer to leave the burden of performing the actual calculation to my interlocutor. As a result, the semantics of an arithmetic expression is the number it evaluates to. How to define *precisely* this evaluation process?

A possible answer could be "perform the leftmost addition between two syntactic numbers and continue until only a single number is left". To turn this mere intuition into a mathematical definition, amounts to defining an **operational semantics**. The operational semantics explains, how an (abstract) machine would go about computing the value of an expression, step-by-step. Formally, it takes the form of a relation $\to$ between expressions describing one step of the machine. For instance $\underline{2}+\underline{2} \to \underline{4}$ indicates that in one step $\underline{2}+\underline{2}$ reduces to $\underline{4}$ whereas $(\underline{1}+\underline{2})+\underline{1} \to \underline{3}+\underline{1} \to \underline{4}$ indicates that $(\underline{1}+\underline{2})+\underline{1}$ reduces in two steps to $\underline{4}$ by going through the intermediate expression $\underline{3}+\underline{1}$. This can be extended into a mathematical definition of $\to$.

An expression e can always be reduced according to $\to$ until an expression of the form $\underline{n}$ is reached (which, then, cannot be further reduced). In this case, the number n is the **semantics** (or **interpretation**) of e and we write $[\![e]\!] = n$.

Variable and functions. Consider now arithmetic expressions that can contain variables: "$x+\underline{3}$" becomes a valid expression $e(x)$ – we indicate the variables occuring inside an expression with the usual mathematical notation. What is its meaning? Clearly, there is nothing to compute, yet this is not a number: we cannot conclude without knowing the value for x. However, if we do know its value, then we can compute and deduce the result. It seems natural to consider that when writing $x+\underline{3}$, what is meant is the *function* mapping a number n to the number $n+3$. Or, in a more mathematical notation, $[\![x+3]\!]$ is $n \mapsto n+3$.

However, to go from an arbitrary expression e to the corresponding function, operational semantics cannot really be extended. A solution would be to say that an expression $e(x)$ has meaning $n \mapsto [\![e(\underline{n})]\!]$ where $\underline{n}$ is the mathematical expression reduced to a number whose value is that of n.

However, we lost the operational flavour: the evaluation of variables is invisible in the semantics. In particular, the expressions $\underline{2} \times x$ and $x + x$ both denote the function that maps a number to its double. We lost *intensional* information about the expression: how the value is computed. In particular $x + x$ has two occurrences of the variable x whereas $2 \times x$ has only one. This loss of intensional information can cause problems when moving to a richer setting, for instance to evaluate expressions $e(x)$ where an occurrence of x yields the result of a coin toss (0 or 1). In this case, $e(x) := 2 \times x$ *always* evaluates to an even number whereas $e'(x) := x + x$ may not, as x could evaluate once to zero, and once to one.

Program/Context dialogue. To account operationally for variables, the operational semantics needs to be updated. Previously, the relation $\to$ only described steps of *internal computation*. We wish to update this vision to allow the context (or the environment) to communicate with the program, in order to provide it with values for the variables: the expression now exchanges messages with its context. A *possible* execution of $x + 2$ is now:

$$x + \underline{2} \xrightarrow{\mathrm{q}_x^+} [] + \underline{2} \xrightarrow{2^-} \underline{2} + \underline{2} \to 4$$

Some steps are now labelled: they denote the messages sent or received by the program during the step. The polarity in superscript indicates whether the message is sent ($+$) or received ($-$) by the program. The first step, labelled q_x^+, is a question to the context: the expression asks the value for x. The expression now awaits an answer from the context, symbolized by the placeholder $[]$. Then, in a second step, the program receives an answer: this occurrence is equal to 2. The placeholder is replaced with the value. Since the resulting expression does not contain any variables, it can simply be computed without communication with the context. The expression $x + 2$ has many possible such executions as the context is free to choose the answer to the question q_x^+.

This solves the problem shown earlier, as $x + x$ has the following execution:

$$x + x \xrightarrow{\mathrm{q}_x^+} [] + x \xrightarrow{0^-} \underline{0} + x \xrightarrow{\mathrm{q}_x^+} 0 + [] \xrightarrow{1^-} \underline{0} + \underline{1} \to 1.$$

If we forget the intermediate programs and internal steps, and simply consider the sequence of messages exchanged, we get: $\mathrm{q}_x^+ \cdot 0^- \cdot \mathrm{q}_x^+ \cdot 1^-$. To remember the final value of the program, we add an extra message at the end where the program signals to the context its result, and also an initial message from the environment to start the computation, which gives the following **dialogue**:

$$\mathrm{q}^- \cdot \mathrm{q}_x^+ \cdot 0^- \cdot \mathrm{q}_x^+ \cdot 1^- \cdot 1^+.$$

Intuitively, a particular dialogue captures the interaction of a particular program *against* a particular context. The semantics of an expression now becomes the set of dialogues where the program interacts against a particular context, *eg.*

$$[\![x + x]\!] = \{\mathrm{q}^- \cdot \mathrm{q}_x^+ \cdot n^- \cdot \mathrm{q}_x^+ \cdot m^- \cdot (n+m)^+ \mid n, m \in \mathbb{N}\}$$

Game semantics and composition. Game semantics makes formal the idea of interpreting programs by dialogues, and define such an interpretation for a variety of programming languages. The main advantage is to be able to interpret programs with unknown variables (that can be of any kind, as in *eg.* the program $f(\underline{3}) + \underline{4}$).

In game semantics, the previous diagram is written as follows:

$$\begin{array}{lcl} (x : \mathbb{N}) & \Rightarrow & \mathbb{N} \\ & & \mathrm{q}^- \\ \mathrm{q}^+ & & \\ 0^- & & \\ \mathrm{q}^+ & & \\ 1^- & & \\ & & 1^+ \end{array}$$

Times flows from top to bottom; the right column (result component) denotes messages about the result and the left column messages about x (x component).

In game semantics, programs are interpreted as sets of *valid dialogues*. Validity of dialogues is defined via a (2-player) **game**. The moves of a game represent the possible messages the program can send or receive (depending on the polarity of the move). Valid dialogues are defined as valid plays of the game. A set of valid dialogues on a game can then be seen as a **strategy** on this game that explains how Player (the program) reacts to Opponent (the context) moves.

This representation makes it easy to represent *composition*. Consider an expression $e(x)$ (say $x + x$) in which another expression $e'(y)$ (say $3 \times y$) is to be plugged for x, resulting in an expression $e(e'(y))$ (in this case, $3 \times y + 3 \times y$). We are interested in deriving the dialogues of $[\![e(e'(y))]\!]$ from those of $[\![e'(y)]\!]$ and $[\![e(x)]\!]$. First, we can form *interaction dialogues* where $e(x)$ is run so that messages sent to the x component are forwarded to $e'(y)$ on its result component and conversely.

An example of such an interaction dialogue is as follows:

$$\begin{array}{lcl@{\qquad}lcl} (y : \mathbb{N}) & \overset{e'(y)}{\Longrightarrow} & \mathbb{N} & (x : \mathbb{N}) & \overset{e(x)}{\Longrightarrow} & \mathbb{N} \\ & & & & & \mathrm{q}^- \\ & & \mathrm{q}^- & & \mathrm{q}^+ & \\ \mathrm{q}^+ & & & & & \\ 1^- & & & & & \\ & & 3^+ & & 3^- & \\ & & \mathrm{q}^- & & \mathrm{q}^+ & \\ \mathrm{q}^+ & & & & & \\ 2^- & & & & & \\ & & 6^+ & & 6^- & \\ & & & & & 9^+ \end{array}$$

Observe that the result component of $e'(y)$ plays *against* the x component of $e(x)$. As a result, the *external* context can only communicate on the y component of $e'(y)$ and the result component of $e(x)$. If we hide the communication in the middle between $e(x)$ and $e'(y)$, we have the following dialogue:

$$\begin{array}{lll} (y : \mathbb{N}) & \Rightarrow & \mathbb{N} \\ & & q^- \\ q^+ & & \\ 1^- & & \\ q^+ & & \\ 2^- & & \\ & & 9^+ \end{array}$$

which is a valid dialogue in the semantics of $3 \times y + 3 \times y = e(e'(y))$, as expected.

This shift from *extensional models* (representing open terms by functions) to *interactive models* (representing open terms by dialogues) was initiated by the question of capturing observational equivalence of pure functional programs.

Compositionality, and observational equivalence. The previous interpretation in terms of dialogues is *compositional*. As seen above, dialogues can be composed via interaction, hence the dialogues for a large expression can be obtained from dialogues of its subparts. This property is interesting from a practical point of view as it makes the semantics easier to compute for a large code base. Since it can be done incrementally, if a little part of the code changes, it is easier to recompute the total semantics since the semantics of the rest need not be recomputed.

However, this property is also interesting from a theoretical standpoint. If two programs have the same set of dialogues, then the programs will have the same behaviour *in any context*: they are **observationally equivalent**. This allows to substitute one for the other in a larger code base without breaking things. The semantics gives a *sound* tool to reason about observational equivalence.

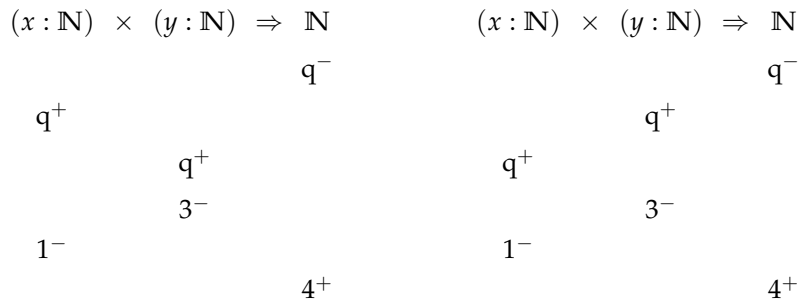
When the converse holds, that is when two programs are observationally equivalent, then they have the same semantics (in our case, the same dialogues), then the interpretation is **fully-abstract**: the semantics provides a *complete* tool to reason about program equivalence. Game semantics originated in the quest for a fully-abstract model of PCF, a pure functional language, since it was shown that its natural extensional interpretation was not fully-abstract [**Plo77**], as the considered functions could represent behaviour lying outside the expressivity of PCF.

The first interactive model was that of sequential algorithms [**BC82**], which still captures only an extension of PCF with a control operator, `catch` [**Cur92**]. Independently, two similar interactive models developed: one by Hyland-Ong [**HO00**] and the other by Abramsky-Jagadesan-Malacaria [**AJM00**]. Through an undecidable quotient, both characterize the observational equivalence for PCF. The models are said to be **intensionally fully-abstract**: observational equivalence in the syntax and in the model coincide. (There is no hope of doing better as observational equivalence for finitary fragments of PCF is undecidable [**Loa01**].)

However their value lies in supporting a wide variety of extensions. In particular, they support extensions with state and control operators, for which these

models give fully abstract models (without quotient): they characterize concretely the observational equivalence for those enriched languages. Moreover, these extensions with computational effects (state, control operators) can be linked to conditions on the shape of strategies (innocence, well-bracketing): a strategy satisfies a condition if and only if its behaviour comes from a program which does not use the corresponding effect. Chapter 5 discusses these extensions in more details.

Concurrency and alternation. So far, the operational model described is **sequential**: operations are done in a linear order, one after the other. What happens if we want to represent parallel evaluation? For instance $x + y$ could be run by asking the value for x and y in parallel, waiting for both answers and then returning the final value. However, currently there is a hidden assumption about our dialogues: they are all **alternating**. Players (the program or the context) takes turns in sending messages. To be able to represent dialogues exhibiting concurrency, we need to relax this assumption and authorize to send a message to the context *before* receiving an answer to the previous message. For instance, the following diagrams depicts dialogues for the expression $x + y$ that feature parallelism:



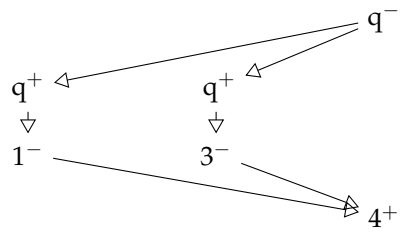
Here $x + y$ interacts against a context that supplies $y = 3$ and $x = 1$. The questions are asked in a non sequential way: both questions are actually sent before receiving any answers. However, the dialogues differ by the order in which the questions are sent (left: x then y; right: y then x). As a result, such dialogues implicitly feature a scheduler resolving the parallelism of the program. Hence a dialogue now represents the interaction of a particular program against a particular context, scheduled in a particular way.

Causal representation of concurrency. This representation of concurrency by linear dialogues is not very satisfactory as dialogues need to contain scheduling information. This leads to a combinatorial explosion since concurrency is essentially represented by interleavings: the number of dialogues used to represent a particular program grows exponentially with the number of parallel computations. Moreover, it is also not satisfactory from a theoretical standpoint: the scheduling information obfuscates the intention of the program and makes it hard for the representation to scale to richer programming settings such as probabilistic programming. Moreover, it makes the correspondence between shapes of dialogues and programming features difficult to generalize.

To work around this problem, it is necessary to take the question of the mathematical representation of concurrency seriously. Insisting to observe the order in which the parallel requests are made, forces the scheduler to *also* be observable.

To relax this assumption, it becomes necessary to come to terms with the impossibility of observing the order in which some messages are sent (or received). In concrete terms, this means moving away from *chronology* to *causality*. Dialogues should not be totally ordered anymore; but only partially-ordered. Two messages are concurrent if they are not comparable for the partial order, ie. we do not know in which order they appear. The two previous linear dialogues can be subsumed by a single partially-ordered diagram:

$$(x : \mathbb{N}) \quad \times \quad (y : \mathbb{N}) \quad \Rightarrow \quad \mathbb{N}$$



Such approaches are often called *truly concurrent* as they represent concurrency as a primitive notion, distinct from the interleaving of (sequential) traces.

Nondeterminism. One last challenge we would like to address is nondeterminism. A program is nondeterministic when it may evaluate to more than one result. Typical examples of nondeterministic programs include primitives to generate pseudo-random numbers that are found in most programming languages.[α] Nondeterminism is also a natural byproduct of shared memory concurrency as races between memory accesses create nondeterministic behaviours.

Suppose that in our language of expressions, we add the construct `choice` which evaluates to zero or to one. The semantics of `choice` in terms of dialogues is very easy to define: it contains the following two dialogues:

$$\begin{array}{cc} \mathbb{N} & \mathbb{N} \\ \\ \mathrm{q}^- & \mathrm{q}^- \\ \downarrow & \downarrow \\ 0^+ & 1^+ \end{array}$$

Each dialogue indicates a possible return value.

In a nondeterministic setting, the question of convergence becomes more subtle. Since a program – even one that does not communicate with the context – is subject to evaluate in different ways, some ways may lead to a final value (converge) while some might be stuck in an infinite loop. For instance, consider the term $\texttt{if}\,(\texttt{choice} = 0)\,\texttt{then}\,0\,\texttt{else}\,\bot$ where $\bot$ represents an infinite loop. This program makes a nondeterministic choice, and based on the outcome, decides to converge or not. However, if we look at the dialogues possible for this program, and the simple program 0, they are the same. Both have the dialogue $\mathrm{q}^- \cdot 0^+$, but the

[α] In practice most pseudo-random generators are deterministic, but it is *apparent* nondeterminism in the sense that the result often depends on parameters outside the control of the programmer, such as the time of execution for example.

former program might diverge whereas the latter will always converge. By only recording the communication with the environment, we have forgotten some information about the program that could be crucial (*eg.* that the program does not *always* terminate). We refer to those behaviours as *hidden divergences*.

Contribution of the thesis. This thesis addresses the issue of turning this intuition of partially-ordered dialogues into a framework for game semantics of concurrent and nondeterministic programming languages. The thesis rests on [**RW11**] that introduces a setting where strategies are **event structures** – a nicer representation of sets of partially-ordered dialogues. However, this work does not address two issues: (1) hidden divergences are not accounted for and (2) only *affine* programs – where variables are used at most once – can be given meaning to.

Plan of the thesis. The thesis splits in three part.

Part 1, construction of the framework. In this first part, we introduce a game semantics framework based on event structures. The part culminates in building a cartesian-closed category on which later developments rest. Moreover, the concurrent interpretation of a simple and nondeterministic language is defined and proved adequate for two notions of convergence (may and must), to give a taste of the expressivity of the model.

Chapter 2: We introduce the basic setting of game semantics based on event structures, following [**RW11, CCRW**]. We then propose an extension of this setting with *essential events* that allow us to track hidden divergences, a key feature in order to model faithfully nondeterministic languages for must convergence. The chapter culminates on the construction of a compact-closed category $\mathrm{CG}^{\cong}_{\circledcirc}$ of games and strategies, both being event structures.

Chapter 3: In $\mathrm{CG}^{\cong}_{\circledcirc}$, strategies cannot play several times the same move, making the model unable to interpret faithfully non affine programming languages (where a variable can occur more than once). To overcome this problem, we follow the AJM [**AJM00**] way by adding copy indices, allowing strategies to play several times the same move. To recover the equations necessary to have a sound interpretation of the λ-calculus, it is necessary to gloss over the exact choice of copy indices of strategies. To this end, we adjoin to event structures a *proof-relevant* equivalence relation, turning them into **event structures with symmetry**. The chapter generalizes the work of Chapter 2 to this setting, by building a compact-closed category $\sim$-$\mathrm{tCG}^{\cong\!\cong}_{\circledcirc}$ of games and strategies as event structures with symmetry.

Chapter 4: This chapter carves out a cartesian-closed category within $\sim$-$\mathrm{tCG}^{\cong\!\cong}_{\circledcirc}$, CHO. It is obtained by consider games of a certain shape that have enough space to accommodate nonlinearity. In this category, we present two interpretations of nondeterministic PCF, one concurrent and one sequential, which are proved adequate for may and must convergences.

Part 2, causal investigations of pure programs. In this second part, we generalize the notions of well-bracketing and innocence traditional in HO game semantics [**HO00**], in order to understand what properties strategies coming from a language without control operators (leading to the notion of *well-bracketing*) or state (leading to *innocence*) satisfy. We finally prove that the interpretations defined in Chapter 4

of **ndPCF** inside the class of well-bracketed and innocent strategies are intensionally fully abstract.

Chapter 5: This chapter introduces our conditions of well-bracketing and innocence, and proves that they are stable under composition so that we get sub-cartesian closed categories of CHO consisting in the innocent, and well-bracketed strategies. Moreover, in this chapter we show a very important property of *visible* strategies (a property weaker than innocence): their interaction is deadlock-free. This means that composition of visible strategies is relational.

Chapter 6: This chapter proves that our interpretations of **ndPCF** given in Chapter 4 are intensionally fully abstract (for may testing). In the process, key properties of innocent and well-bracketed properties are investigated. In particular, we show that innocent strategies support a reduced form which generalizes the P-view tree of strategies in HO games. This induces a notion of *finite* strategy. We also show that innocent and well-bracketed strategies on a higher-order type can be decomposed into smaller strategies of higher-order type and a strategy of first-order type. This allows us to reduce finite definability to finite definability at first-order types.

Part 3, specification of weak memory models.

Chapter 7: In this chapter, we give inside event structures, a model of a simple assembly language abiding by the TSO specification. Because the language is first-order, there is no need for the game semantics machinery to give an interactive and accurate model of it. The chapter defines several models that try to exploit as much as possible the expressive power of event structures to build more concurrent (and more compact) models.

Chapter 8: We recast the constructions used on event structures in Chapter 7 inside our game semantics framework. We show that using strategies and their composition it is possible to recast the model construction of Chapter 7 to get another point of view. This formulation in terms of strategies represents better the execution of programs on relaxed architectures and allows for simple tuning by simply changing the strategies implementing the base operations. This permits scaling in a simpler way to weaker architectures allowing more reorderings.

Part 1

Concurrency

In this first part, we introduce a game semantics framework based on event structures. The part culminates in building a cartesian-closed category on which later developments rest. Moreover, the concurrent interpretation of a simple nondeterministic language is defined and proved adequate for two notions of convergence (may and must), to give a taste of the expressivity of the model.

Outline of the part.

Chapter 2. We introduce the basic setting of game semantics based on event structures, following [**RW11, CCRW**]. We then propose an extension of this setting with *essential events* that allow us to track hidden divergences, a key feature in order to model faithfully nondeterministic languages for must convergence. The chapter culminates on the construction of a compact-closed category $\mathrm{CG}^{\cong}_{\circledcirc}$ of games and strategies, both being event structures.

Chapter 3. In $\mathrm{CG}^{\cong}_{\circledcirc}$, strategies cannot play several times the same move, making the model unable to interpret faithfully non affine programming languages (where a variable can occur more than once). To overcome this problem, we follow the AJM [**AJM00**] way by adding copy indices, allowing strategies to play several times the same move. To recover the equations necessary to have a sound interpretation of the λ-calculus, it is necessary to gloss over the exact choice of copy indices of strategies. To this end, we adjoin to event structures a *proof-relevant* equivalence relation, turning them into **event structures with symmetry**. The chapter generalizes the work of Chapter 2 to this setting, by building a compact-closed category $\sim$-$\mathrm{tCG}^{\cong}_{\circledcirc}$ of games and strategies as event structures with symmetry.

Chapter 4. This chapter carves out a cartesian-closed category within $\sim$-$\mathrm{tCG}^{\cong}_{\circledcirc}$, CHO. It is obtained by consider games of a certain shape that have enough space to accommodate nonlinearity. In this category, we present two interpretations of nondeterministic PCF, one concurrent and one sequential, which are proved adequate for may and must convergences.

CHAPTER 2

Concurrent games with essential events

In Chapter 1, we played around with the idea of interpreting concurrent programs by sets of partially-ordered dialogues, representing *strategies*. Partial orders are used to capture *causality* rather than *chronology*. In this chapter, we detail how to construct a mathematical theory of such strategies. Instead of defining strategies as sets of dialogues, we use **event structures** [**Win86**], that elegantly represent sets of partial orders in one single structure. This aspect is key to remember the nondeterministic branching point of programs. (See Section 1.1.2).

The foundations presented here are based on the framework of [**RW11**], which gives a compact-closed category of strategies up to isomorphism. Because composition is defined by hiding internal events, this model loses track of some divergences occurring during the composition. In this chapter, we extend this framework to remember all divergences, by altering hiding. Our approach here uses *essential events*, in contrast to that of [**CHLW14**] based on stopping configurations. Essential events allow us to retain all the behaviours of strategies before hiding, up to weak bisimulation (Lemma 2.71), which is crucial to model faithfully nondeterministic languages.

Related work. There are several approaches to concurrent game semantics, using different ways of representing concurrent plays. The more conservative approach with respect to sequential game semantics is to remove the alternation hypothesis on traditional dialogues. This approach was explored by Laird [**Lai01**] and later refined by Ghica and Murawski [**GM07**]. Our causal models can be collapsed (by taking the traces of our strategies) to theirs [**CC16**].

Truly concurrent approaches date back to [**AM99b**] (in the deterministic setting) using closure operators, to model a fragment of linear logic. Melliès later considered *asynchronous games* which enrich traditional games with homotopy tiles representing the independence of moves. Strategies are certain sets of alternating paths of these asynchronous transition systems. In a series of work [**Mel03, Mel06, Mel05a**] Melliès exploited this independence information in games to prove crucial properties of his (sequential) strategies (eg. positionality), culminating in a fully complete model of full Linear Logic [**Mel05b**]. Later, Melliès and Mimram [**MM07**] extended this setting to allow non-alternating strategies, with further conditions enforcing the existence of an implicit causal structure. Such strategies correspond to the deterministic fragment of our model [**RW11**].

The first work on using explicit causal structure was in the context of ludics with Faggian and Maurel's *ludics nets* [**FM05**] whose link with game semantics was studied by Curien and Faggian [**CF05**]. Later, Faggian and Piccolo arrived to a partial-order formulation of strategies [**FP09**], which inspired, and is generalized by [**RW11**], the starting point of this thesis.

In another direction, a more recent approach to truly concurrent game semantics is developed by Hirschowitz *et al.* [**Hir14, EHS13**] where strategies become hyper-graphs represented as presheaves. This approach inspired Ong and Tsukada's model of the nondeterministic λ-calculus [**TO15**].

Outline of the chapter. In Section 1, we introduce event structures, and games and strategies based on them. Section 2 introduces interaction of strategies, which is the key step to define composition of strategies. Section 3 introduces the compositional framework of [**RW11**] based on hiding *all* internal events. This gives a compact-closed category of strategies up to isomorphism. Section 4 introduces a new notion of composition where *no* events are hidden during the interaction. In this setting, we take hidden divergences into account, but to get a compact-closed category, we are forced to consider strategies up to weak bisimulation. To get the best of both worlds, we investigate which events should be hidden during composition so that no divergences are hidden while still leading to a compact-closed category up to isomorphism. This leads to the essential events of Section 5.

Contributions of this chapter. Sections 1, 2, 3 are based on the development of Sylvain Rideau and Glynn Winskel [**RW11**] and presentation is taken from [**CCRW**], joint work with Pierre Clairambault, Sylvain Rideau and Glynn Winskel. Sections 4 5, 6 are joint (unpublished at the time of writing) work with Pierre Clairambault, Jonathan Hayman and Glynn Winskel.

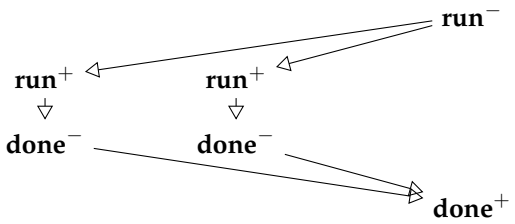
1. Games and strategies as event structures

This section introduces event structures as a model of concurrency and nondeterminism, and presents how to use them as a foundation for games and strategies.

1.1. Towards concurrent strategies. In chapter 1, we presented briefly partially-ordered dialogues as a tool to represent concurrent programs. We now investigate more formally the mathematical content of such diagrams, to finally arrive at a notion of *concurrent strategies* playing on a *concurrent game*.

1.1.1. *Causality.* As a first example of concurrent strategies, consider the `join` primitive that spawns two calculations in parallel and terminates *when* both have terminated. Such a primitive would have the type **proc** $\to$ **proc** $\to$ **proc** where **proc** is the type of commands, that is programs performing computational effects but returning no value of interest. It could be described by the following dialogue:

$$a : \mathbf{proc} \quad \times \quad b : \mathbf{proc} \quad \Rightarrow \quad \texttt{join}\, a\, b : \mathbf{proc}$$



The previous examples of Chapter 1 played on natural numbers where the allowed moves where q (question for the value) and n (answer to the question).

Since on **proc** there are no values to return, the moves are here **run** (beginning of the computation) and **done** (end of the computation).

This diagram represents now a partial order which is generated by the transitive closure of $\rightarrowtriangle$ (**immediate causal dependency**). Note the inversion of polarity between the two sides of $\Rightarrow$. This inversion was already present in the previous chapter and is due to `join` acting as the environment for a and b. When the context runs `join` a b, a and b are run immediately, in any order as the two events $\mathbf{run}^+$ are incomparable. When *both* process terminate (issuing a $\mathbf{done}^-$) the program signals that it has terminated.

In this context, a valid (partial) execution is a set of events which is such that if an event e occurs, then any of its immediate causes $e' \rightarrowtriangle e$ must appear. In other words, an execution is a downclosed subset of events.

1.1.2. *Nondeterminism.* Most concurrent programming features induce nondeterminism: be it shared memory concurrency (à la Concurrent Idealized Algol [**Bro96a**]) or channels (à la CCS [**Mil82**]). To accommodate nondeterminism, causal dependency is however not enough, as nondeterminism implies that two events might be incompatible: for instance the outcomes of a nondeterministic coin-tossing. Nondeterminism is usually modelled using *sets* of executions: in our case, sets of partial-orders. This can work (and is done in [**CC16**]) but it makes nondeterminism "global". There is no notion of *events of the program* simply of *events of an execution*. This point of view is mathematically simple but forgets intensional behaviour of the program, in particular the location of the nondeterministic branching points. This is necessary to distinguish the terms $M_1 = \lambda f.\, f\,\mathrm{tt} + f\,\mathrm{ff}$ and $M_2 = \lambda f.\, f(\mathrm{tt} + \mathrm{ff})$ (where $+$ is nondeterministic choice), which are usually identified models based on set of plays, albeit not must-equivalent.[α]

With sets of executions, we only see where the executions differ, but not the exact point where the program actually made the choice. Moreover, having a notion of *events of the program* turns out to be very handy when defining *essential events*.

Incompatiblity of events is represented via *conflict*, which has two common formal representations in this partial order setting:

- **Binary conflict**: the partial order structure is enriched with a binary relation representing when two events are in conflict, that is when they cannot occur together in a valid execution.
- **General conflict**: the partial order structure is in this case enriched with a set of *consistent sets* formalizing which sets of events can occur together: two events can be conflicting only in a particular context (ie. $x \cup \{e_i\}$ can be consistent but $x \cup \{e_1, e_2\}$ might not even if $\{e_1, e_2\}$ is consistent).

To develop our framework (this chapter and Chapter 3), we use *general conflict* because the mathematical theory is simpler, even though the programming languages considered in this thesis only express binary conflict. From Chapter 4, we will use binary conflict as it is simpler to manipulate concretely. Note that an event structure with general conflict can be be unfolded to a (bisimilar) event structure with binary conflict[**vGP09**].

[α] Consider the context $C[] = []\,(\lambda x.\, \mathtt{if}\ x\ (\mathtt{if}\ x\ \mathrm{tt}\ \bot)\ \mathrm{tt})$, $C[M_1]$ must converge (since in each execution x is true or false), but $C[M_2]$ must not.

DEFINITION 2.1 (Event structures (with general conflict)). An **event structure** is a triple $(E, \leq, \mathrm{Con}_E)$ where $(E, \leq_E)$ is a partial order of events and $\mathrm{Con}_E \subseteq \mathscr{P}_f(E)$ is a set of *finite* **consistent** sets of E subject to the following axioms:

(1) For every $e \in E$, the set $[e] = \{e' \in E \mid e' \leq e\}$ is finite,
(2) For all $Y \in \mathrm{Con}_E$ and $X \subseteq Y$ then $X \in \mathrm{Con}_E$,
(3) For all $e \in E$, $\{e\} \in \mathrm{Con}_E$,
(4) If X is consistent, so is its downclosure

$$[X]_E = \{e' \in E \mid e' \leq e \text{ for some } e \in X\}.$$

In particular, if $X \in \mathrm{Con}_E$ and $e \leq e' \in X$ then $X \cup \{e\} \in \mathrm{Con}_E$.

Binary conflict is defined as a special case of general conflict:

DEFINITION 2.2 (Binary conflict). An event structure S has binary conflict if there exists a (necessarily unique) symmetric binary relation $\sharp_S \subseteq S^2$ such that

$$\mathrm{Con}_S = \{X \text{ finite} \mid X^2 \cap \sharp_S = \emptyset\}.$$

A binary conflict $s \sharp_S s'$ is said to be **minimal** when the set $\{s_0 \in S \mid s_0 < s \vee s_0 < s'\}$ is consistent in S. In that case, we write $s \rightsquigarrow s'$.

1.1.3. *Notations on event structures.* Given an event structure E, write $e \rightarrowtriangle_E e'$ for **immediate causality** defined as $e < e'$ with no events in between. If $e \leq_E e'$, we say that e' **causally depend** on e. If $e \rightarrowtriangle_E e'$, we say that e' **immediately causally depends** on e, or that there is a **causal link** from e to e'. Given $e \in E$, write $[e) = [e] \setminus \{e\}$. Two events $e, e' \in E$ are **concurrent** when they are incomparable for the causal order and $\{e, e'\} \in \mathrm{Con}_E$.

Event structures induce a notion of execution through configurations:

DEFINITION 2.3 (Configuration). A **configuration** of an event structure E is a subset x of E such that:

- x is downclosed (ie. $e' \leq e$ and $e \in x$ imply $e' \in x$),
- all finite subsets of x are consistent.

The notion of configurations is crucial when reasoning on event structures: most proofs will be carried out at the level of configurations. As a result, it will be important to understand, for each construction defined at the level of event structures, its action at the level of configurations.

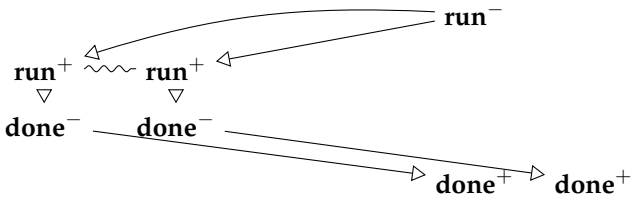
The set of *finite* configurations of E will be written $\mathscr{C}(E)$. Configurations are naturally ordered by inclusion. Moreover, any configuration inherits a partial order from E. A configuration $x \in \mathscr{C}(E)$ **extends** by $e \in E$ (written $x \overset{e}{\text{—}\!\!\subset} x \cup \{e\}$) when $e \notin x$ and $x \cup \{e\} \in \mathscr{C}(E)$. In that case, e is called an **extension** of x. Two extensions of e, e' of x are **compatible** when $x \cup \{e, e'\} \in \mathscr{C}(E)$, **incompatible** otherwise. In that case, we say that e and e' are a **minimal conflict** in the context x (or **involved in a minimal conflict**). In the general case, it depends on the context x, but when the event structure has binary conflict, it is independent from x and coincide with the notion of minimal conflict introduced above.

A consequence of the axioms of event structures is that for every $e \in E$, the sets $[e]$ is a configuration representing the causal history of e. Such configurations are called **prime configurations**, or equivalently a prime configuration is a configuration with a top element. Remark that, consequently $[e)$ is also always a configuration. Given a configuration x of an event structure E, a **covering chain** is a sequence $\emptyset = x_0 \text{—}\!\subset x_1 \text{—}\!\subset \ldots \text{—}\!\subset x_n = x$ of configurations leading to x.

1.1.4. *Drawing event structures.* Pictures will only feature event structures with binary conflict and represent immediate causality ($\rightarrowtriangle$) and minimal conflict ($\rightsquigarrow$).

EXAMPLE 2.4. The interpretation of the nondeterministic sum of processes can be represented by the event structure:

$$a : \mathbf{proc}, \quad b : \mathbf{proc} \quad \vdash \quad \mathtt{sum}\, a\, b \quad : \mathbf{proc}$$



The difference with the `join` operator is the conflict between the two occurrences of $\mathbf{run}^+$ that ensures the presence of only one of the $\mathbf{run}^+$ in a single execution. The conflict propagates upwards: the causal future of these events are also in conflict. In this example, the names **run**, **done** come from an (implicit) labelling of events by moves of a game. Such labelled event structures will be used to represent strategies on games.

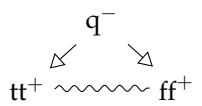
1.2. Games and pre-strategies. We now define a notion of games and pre-strategies on them to make formal the diagrams of the previous section. Throughout this thesis, we will introduce several notions of "strategies" and "pre-strategies". The word "pre-strategy" is used to refer to objects supporting a notion of (associative) composition – but nothing more. "Strategies" is used to refer to the class of pre-strategies that is invariant under composition by a particular strategy, copycat, taken to be the identity: strategies naturally organize themselves into a category.

1.2.1. *Games as event structures with polarities.* In our setting, games will simply be event structures where each event carries a polarity:

DEFINITION 2.5 (Game). A **game** (or **event structure with total polarities**) is an event structure E along with a polarity labelling $\mathrm{pol}_E : E \to \{+, -\}$.

Event structures with polarities will be drawn as event structures, where polarity is indicated in superscript of events. We will also make use of the notation "let $a^+ \in A$" to introduce a positive event of a game A (similarly "let $a^- \in A$" to introduce a negative event). We will often use the term **play** to refer to a configuration of a game by analogy with the standard game-theoretic terminology.

A possible game for booleans $\mathbf{B}$ is as follows:



The initial question denotes the call from the environment and the two positive moves denote the possible return values of the program.

Rules of a game are specified via its causal order and consistent sets:

- *causal order*: a move cannot be played before another is played,
- *consistency*: some moves cannot occur together (e.g., for booleans, the moves corresponding to true and false cannot be played together).

1.2.2. *Operation on games.* To build games, we will make use of two fundamental operations on games: duality and parallel composition. Given a game A, the **dual game** $A^\perp$ is simply obtained by exchanging polarities, leaving the event structure untouched. The **simple parallel composition**, or more briefly **parallel composition**, of A and B, denoted by $A \parallel B$ is defined as follows:

DEFINITION 2.6 (Simple parallel composition). Let A_0 and A_1 be event structures. The event structure $A_0 \parallel A_1$ is defined as follows:

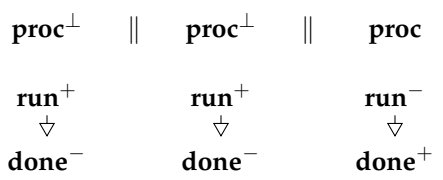
Events: $\{0\} \times A_0 \cup \{1\} \times A_1$
Causality: $(i,a) \leq_{A_0 \parallel A_1} (j,a')$ iff $i = j$ and $a \leq_{A_i} a'$
Consistency: $X \in \mathrm{Con}_{A_0 \parallel A_1}$ iff $\{a \mid (i,a) \in X\} \in \mathrm{Con}_{A_i}$ for $i \in \{0,1\}$

Moreover, any choice of polarities on A_0 and A_1 induce a canonical choice of polarities on $A_0 \parallel A_1$ via $\mathrm{pol}_{A_0 \parallel A_1}(i,a) = \mathrm{pol}_{A_i}(a)$.

Parallel composition will be used to interpret product types. In $A \parallel B$, A and B evolve concurrently with no interference (no causality or conflict). As a result, the following monotonic map is an order-isomorphism:

$$\begin{aligned} \cdot \parallel \cdot : \mathscr{C}(A) \times \mathscr{C}(B) &\to \mathscr{C}(A \parallel B) \\ (x,y) &\mapsto x \parallel y = \{0\} \times x \cup \{1\} \times y \end{aligned}$$

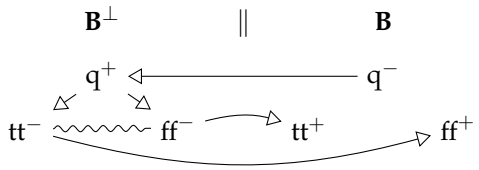
These operations can be used to represent the game on which `join` and `sum` play, as the game $\mathbf{proc}^\perp \parallel \mathbf{proc}^\perp \parallel \mathbf{proc}$:



1.2.3. *Pre-strategies.* The strategies given in Section 1.1 can be viewed as a causal and conflict enrichment of the game. Such strategies on a game A can be represented as event structures $(S, \leq_S, \mathrm{Con}_S)$ where $S \subseteq A$, $\leq_S \supseteq \leq_A$ (causal enrichment) and $\mathrm{Con}_S \subseteq \mathrm{Con}_A$ (conflict enrichment). This captures the requirement that strategies must respect the rules of the game A, that is:

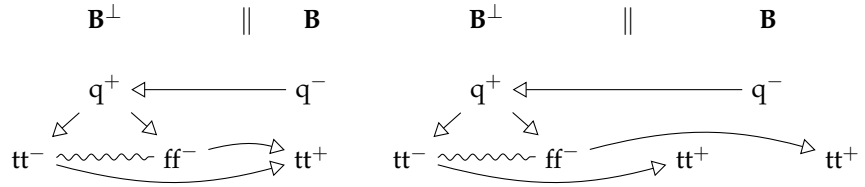
- play a move only after all the moves it depends on in the game occurred,
- consistent sets of moves for strategies are also consistent in the game.

EXAMPLE 2.7. Boolean negation can be represented as a strategy on $\mathbf{B}^\perp \parallel \mathbf{B}$.



Note that the arrows between the question and the answers on $\mathbf{B}$ (the right component) are induced by transitivity, and the inconsistency between the positive true and false is also induced by the inconsistency on their negative counterpart.

However, how to define the boolean function that *evaluates its argument* and then returns true in both cases? Two candidate diagrams come to mind:



The left diagram has one occurrence of tt^+ that depends on the two possible argument values; this is not valid because the two Opponent moves are in conflict: the downclosure of $\{\mathrm{tt}^+\}$ is not consistent. The second solution describes a valid event structure but has two tt moves: it is not a subset of the game anymore. Note that the `sum` strategy given in Example 2.4 is also not a proper subset of the game $\mathbf{proc}^\perp \parallel \mathbf{proc}^\perp \parallel \mathbf{proc}$ as $\mathbf{done}^+$ has two occurrences. As a result, to account for these behaviours, strategies should not be strict subsets of the game, but *embeddings*. The notion of embedding is formalized via maps of event structures:

DEFINITION 2.8 (Maps of event structures). A (total) **map of event structures** from E to F is a function on events $f : E \to F$ such that:

- the direct image of a configuration of E is a configuration of F,
- f is injective on consistent sets.

Event structures and their (total) maps form a category $\mathcal{E}$.

A **pre-strategy** on a game A will be a map of event structures $\sigma : S \to A$. In that case, we write $\sigma : A$ to denote that σ is a pre-strategy on A. The first condition makes sure that the plays of S are valid according to A whereas the second one is a linearity condition ensuring that in a play, events of the game occur at most once. This linearity condition will be crucial to define interaction of pre-strategies in Section 2. If $\sigma : S \to A$ is a pre-strategy, we will sometimes write $\mathscr{C}(\sigma)$ for $\mathscr{C}(S)$.

According to this definition, pre-strategies $\sigma : S \to A$ appear as certain event structures labelled by events of the game. As a consequence, events of a pre-strategy (events of S) naturally carry a polarity given by the labelling $\mathrm{pol}_A \circ \sigma$, and S can be regarded as an event structure with polarities. A pre-strategy on A is also a pre-strategy on $A^\perp$ since the definition is independent from the polarity on A. We now move on to a very important example of pre-strategies.

1.2.4. *The copycat pre-strategy.* In the game $A^\perp \parallel A$, each move of A appears twice, with a different polarity. A natural strategy on this game is as follows: to play the positive occurrence of $a \in A$, we wait for its negative counterpart on the other side. This describes the **copycat** pre-strategy. It is implemented by adding a causal link from the negative occurrence of every $a \in A$ to its positive occurrence.

DEFINITION 2.9 (Copycat strategy). Let A be a game. The copycat pre-strategy $\mathfrak{a}_A$ on $A^\perp \parallel A$ is defined as the identity-on-events map:

$$\mathfrak{a}_A : \mathfrak{CC}_A \to A^\perp \parallel A,$$

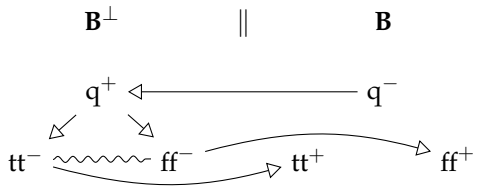
where $\mathfrak{CC}_A$ is given by:

Events: $A^\perp \parallel A$
Causality: Transitive closure of

$$\leq_{A^\perp \parallel A} \cup \{((i,a),(1-i,a)) \mid (i,a) \text{ negative in } A^\perp \parallel A\}$$

Consistency: $X \in \mathrm{Con}_{\mathfrak{CC}_A}$ iff $[X]_{\mathfrak{CC}_A} \in \mathrm{Con}_{A^\perp \parallel A}$.

As an example, the copycat strategy on the game **B** is given by:



The definition of $\leq_{\mathbb{CC}_A}$ is convoluted, and we do not have a direct grasp on the immediate causality of $\mathbb{CC}_A$, issue addressed by the following lemma:

LEMMA 2.10. *For $a, a' \in A$, there is an immediate causality $(i,a) \rightarrowtriangle_{\mathbb{CC}_A} (j,a')$ if and only if one of the following two conditions holds:*

(1) $i = j$, $a \rightarrowtriangle_A a'$ and either (i,a) is positive or (j,a') negative in $\mathbb{CC}_A$.
(2) $i \neq j$, $a = a'$, and $(i,a) \in \mathbb{CC}_A$ is negative.

Case (1) describes causal links inherited from the game (never of the shape $- \rightarrowtriangle +$) and (2) describes causal links between $A^\perp$ and A (always of this shape).

PROOF. It is clear that both conditions imply $(i,a) \rightarrowtriangle_{\mathbb{CC}_A} (j,a')$. Conversely, we know $\leq_{\mathbb{CC}_A}$ is generated by $\rightarrowtriangle_{A^\perp \| A} \cup \{((i,a),(1-i,a) \mid (i,a)^- \in \mathbb{CC}_A\}$. This means that $(i,a) \rightarrowtriangle (j,a')$ implies either $i \neq j$, $a = a'$ and $(i,a)^- \in \mathbb{CC}_A$ (as desired) or $i = j$ and $a \rightarrowtriangle_A a'$. In this case, if (i,a) is negative and (j,a') is positive, we have $(i,a) \rightarrowtriangle_{\mathbb{CC}_A} (1-i,a) <_{\mathbb{CC}_A} (1-i,a') \rightarrowtriangle_{\mathbb{CC}_A} (i,a')$ contradicting $(i,a) \rightarrowtriangle_{\mathbb{CC}_A} (j,a')$. Hence (i,a) is positive or (j,a') is negative. □

Configurations of copycat also have a simple description that highlights a very important structure of the domain of configurations of a game.

LEMMA 2.11. *A configuration $x \parallel y \in \mathscr{C}(A^\perp \parallel A)$ is a configuration of $\mathbb{CC}_A$ if and only if $x \supseteq^+ x \cap y \subseteq^- y$ where $x \cap y \subseteq^+ x$ means $x \cap y \subseteq x$ and all events of $x \setminus x \cap y$ are positive and similarly for $x \cap y \subseteq^- y$.*

The induced relation on configurations of A: $x \sqsubseteq_A y$ if and only if $x \supseteq^- x \cap y \subseteq^+ y$ is a partial order called the **Scott order**. This partial order plays a crucial role in the proof of Theorem 2.38 presented in **[CCRW]**.

1.2.5. *Properties of maps.* We end this section by a little discussion on maps of event structures and their properties, that will be useful later. First, even though maps do not preserve causality, they always *reflect* it in a certain sense:

LEMMA 2.12. *Let $f : A \to B$ be a map of event structures and $a, b \in A$ such that $\{a,b\}$ is consistent. If $f(a) \leq f(b)$ then $a \leq b$.*

PROOF. Since f is a map of event structures, $f[b]$ is downclosed as a configuration of F. Since $f(a) \leq f(b) \in f[b]$ by hypothesis, it follows that $f(a) \in f[b]$ and thus $f(a) = f(c)$ for some $c \leq b$. Since $\{a,b\}$ is consistent so is $\{a,b,c\} \subseteq [\{a,b\}]$ and local injectivity implies $a = c \leq b$ as desired. □

Moreover, any map of event structures $f : A \to B$ induces a monotonic map $f : \mathscr{C}(A) \to \mathscr{C}(B)$. This map completely characterizes the behaviour of f on events, which allows to prove equality of maps at the level of configurations:

LEMMA 2.13. *Let $f, g : A \to B$ be maps of event structures such that for all configuration $x \in \mathscr{C}(A)$ we have $fx = gx$. Then $f = g$.*

PROOF. Let $a \in A$. Remember that $[a) = [a] \setminus \{a\}$ is a configuration of A. By hypothesis we have $f[a] = g[a]$ and $f[a) = g[a)$ as sets, thus $\{f(a)\} = f[a] \setminus f[a) = g[a] \setminus g[a) = \{g(a)\}$ and hence $f(a) = g(a)$. □

Simple parallel compositions extends to maps by letting $f_0 \parallel f_0 : A_0 \parallel A_1 \to B_0 \parallel B_1$ to be $(f_0 \parallel f_0)(i, a) = (i, f_i(a))$ for maps $f_i : A_i \to B_i$. As usual we will often write $f \parallel C : A \parallel C \to B \parallel C$ for the map $f \parallel \mathrm{id}_C$.

2. Closed interaction of pre-strategies

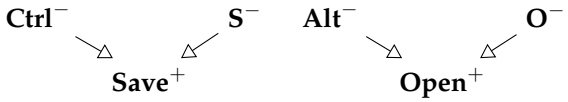
A natural operation on a concept of "strategy" is that of interaction. In our setting, this prompts the question: given a pre-strategy σ on A, and a pre-strategy τ on $A^\perp$, what is their interaction? It can be seen as running σ against τ. Since both σ and τ are concurrent and nondeterministic, this process will be nondeterministic and concurrent. It should exhibit the *common behaviour* of σ and τ: only moves that σ and τ are ready to play should appear, *when* they are both ready to play. The result is an event structure (with no clear polarities) describing this process.

In this section, we omit proofs and refer the reader to **[CCRW]** for details.

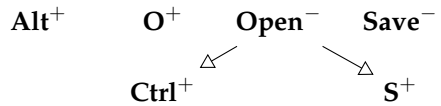
2.1. Examples of interaction. In this subsection, we illustrate what interactions of particular strategies should be, by picking examples outside the realm of programming languages. In these examples, there will be a strategy $\sigma : A$ interacting against $\tau : A^\perp$. The game A will represent a user interface (from the point of view of the user), σ the user interacting with the interface and τ implementing the interface. For the purpose of interaction, the actual structure of the game (consistency and causality) does not matter: games in the examples will be reduced to a set of moves, without causality or non-consistency.

EXAMPLE 2.14 (Causality and interaction). Imagine the interface for an extremely simple text editor where the protocol between the user and the program is modeled by the game $A = \mathbf{Ctrl}^+ \, \mathbf{S}^+ \, \mathbf{Alt}^+ \, \mathbf{S}^+ \, \mathbf{O}^+ \mathbf{Open}^- \mathbf{Save}^-$ describing some actions available in a text editor. The user can press some keys, and the program can open or save a file. The program opens the file as soon as Alt and O are pressed, and saves it when Ctrl and S are pressed.

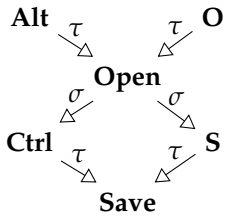
Such a behaviour can be described as the following pre-strategy on $A^\perp$:



In the meantime, imagine that the user wants to open a file and save it. Knowing the magic shortcuts, they press Alt+O, wait for the file to be opened, press Ctrl+S and contemplate their file being saved.



Their interaction proceeds as follows: the first events which are both ready to play are **Alt** and **O** – those are the only initial moves of both σ and τ. After those two moves, the move **Open** becomes available, which in turn enables **Ctrl** and **S** which finally enable **Save**. Hence the resulting interaction is:



In this diagram, the immediate causal dependencies are labelled by the pre-strategy they come from. At the end, we get the (transitive closure of the) union of the two partial orders.

However, this union might not always be a valid partial order:

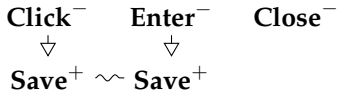
EXAMPLE 2.15 (The boring drug deal). Let $A = \mathbf{Drug}^+ \, \mathbf{Money}^-$, and consider the interaction of $\sigma = \mathbf{Drug}^+ \Leftarrow \mathbf{Money}^- : A$ (the seller) against $\tau = \mathbf{Drug}^- \Rightarrow \mathbf{Money}^+ : A^\perp$ (the buyer). Since they do not have a common minimal event, there is a **deadlock**. As a result, the interaction between σ and τ is empty.

Finally, we need to illustrate one more aspect of interactions, in relation to nondeterminism. In these examples, it appears that events of the interaction can be seen as events of S and T. However, when nondeterminism (and especially non-injectivity of the labelling function) comes in, it is no longer the case.

EXAMPLE 2.16 (Duplication). Consider the following game:

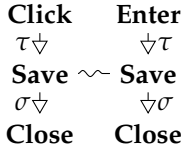
$$A = \mathbf{Click}^+ \, \mathbf{Enter}^+ \, \mathbf{Save}^- \, \mathbf{Close}^+,$$

where the user can click on a button or press enter to save a file; independently, they can also choose to close the editor. The program $\tau : A^\perp$ is given by:



This is an instance of *disjunctive causality*: the file should be saved if enter was pressed **or** the button clicked. This disjunctive causality forces the pre-strategy to be non-injective as the event structures presented here cannot directly express that one event may have several causal histories. More precisely, the labelling map σ is not injective as $\mathbf{Save}^+$ has two occurrences. The user $\sigma : A$ is given by:

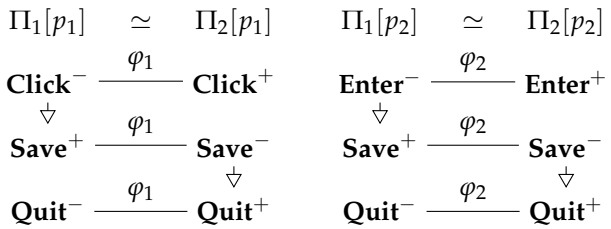
This user clicks and presses enter at the same time. As there are two events for **Save** in τ, in the interaction there must also be two events **Close** because of the causal link $\mathbf{Save} \Rightarrow \mathbf{Close}$ in σ. The interaction is:



On this example, we observe that events do not always correspond to pairs of events of σ and τ, as duplication is propagated upwards. In particular, the two **Close** events of the interaction correspond to the same events in both σ and τ.

It turns out that events of the interaction do not relate in a simple way to events of the strategies. To understand the last example, it is better to look at its configurations (Section 2.2), from which the events can be derived (Section 2.3).

2.2. Interaction states. Write p_1 and p_2 for the two **Close** events of the interaction in Example 2.16. As noticed above, they correspond to the same events both in σ and τ but their causal histories differ: $[p_1]$ contains a **Click** move whereas $[p_2]$ contains a **Enter**. From $[p_1]$ we can extract a configuration of S, written $\Pi_1[p_1]$ and a configuration of T, written $\Pi_2[p_1]$. Similarly, $[p_2]$ induces $\Pi_1[p_2] \in \mathscr{C}(S)$ and $\Pi_2[p_2] \in \mathscr{C}(T)$. Those pairs of configurations have the same image in the game: $\sigma(\Pi_1[p_1]) = \tau(\Pi_2[p_1])$ and likewise for p_2. By local injectivity, it follows that there are bijections $\varphi_1 : \Pi_1[p_1] \simeq \Pi_2[p_1]$ and $\varphi_2 : \Pi_1[p_2] \simeq \Pi_2[p_2]$ described as follows:



2.2.1. *Secured bijections.* In both cases, the orders on both sides are causally compatible (the union is acyclic). Such bijections are called **secured bijections**:

DEFINITION 2.17 (Secured bijection). Let $\mathbf{q}, \mathbf{q}'$ be partial orders and $\varphi : \mathbf{q} \simeq \mathbf{q}'$ be a bijection (non necessarily order-preserving). It is **secured** when one of the two equivalent conditions are met:

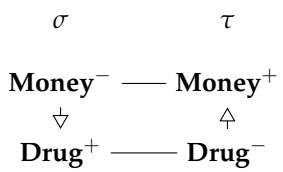
- (*Acyclicity*) The following relation $\lhd_\varphi$ on the graph of φ is acyclic:

$$(s, \varphi(s)) \lhd_\varphi (s', \varphi(s')) \text{ iff } s \rightarrow_{\mathbf{q}} s' \vee \varphi(s) \rightarrow_{\mathbf{q}'} \varphi(s')$$

- (*Inductive construction*) There exists a sequence $e_0, \ldots, e_n$ such that $\mathbf{q} = \{e_0, \ldots, e_n\}$, and for all i, both $\{e_0, \ldots, e_i\}$ and its image by φ are down-closed in $\mathbf{q}$ and $\mathbf{q}'$ respectively.

The partial order, induced by transitive and reflexive closure of $\lhd_\varphi$, is written $\leq_\varphi$.

The acyclicity condition bans causal loops. Indeed, in the drug deal example (Example 2.15), the obvious bijection:



is not secured because of the cycle in the induced preorder.

2.2.2. *Interaction states.* Following the observation of Section 2.2, we use secured bijections to describe the candidate configurations of an interaction.

Let A be a game, $\sigma : S \to A$ and $\tau : T \to A^\perp$ be pre-strategies. If $(x, y) \in \mathscr{C}(S) \times \mathscr{C}(T)$ is such that $\sigma x = \tau y \in \mathscr{C}(A)$, write $\varphi_{x,y}$ for the bijection $x \cong \sigma x = \tau y \cong y$ obtained by local injectivity of σ and τ. The pair (x, y) is an **interaction state** when the corresponding bijection $\varphi_{x,y}$ is secured. We write $\mathscr{S}_{\sigma,\tau}$ for the set of such interaction states, viewed as the set of candidate configurations of a certain event structure. How to reconstruct the corresponding event structure from $\mathscr{S}_{\sigma,\tau}$?

2.3. Prime construction. For an event structure E, the events of E can be recovered from $\mathscr{C}(E)$ by considering those configurations of the form $[e]$ for $e \in E$, called **prime configurations**. Causality and consistency can be understood through this correspondence. However, prime configurations can be equivalently described as configurations with a top-element, which does not refer to events.

This leads to the following definition. Note that since all our definitions regarding the interaction of pre-strategies make sense in a setting without polarity, we state the definition for simple maps of event structures.

DEFINITION 2.18 (Closed interaction of pre-strategies). Let A be an event structure, and $\sigma : S \to A$ and $\tau : T \to A$ be maps of event structures. The following data defines an event structure $S \wedge T$:

Events: Those interaction states $(x, y) \in \mathscr{S}_{\sigma,\tau}$ such that $\varphi_{x,y}$ has a top element (called **prime interaction states**).
Causality: Pairwise inclusion.
Consistency: A finite set of interaction state $X \subseteq S \wedge T$ is consistent iff its pairwise union $\bigcup X$ is an interaction state in $\mathscr{S}_{\sigma,\tau}$.

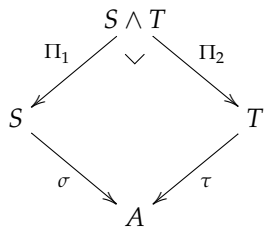
This is an instance of the *prime construction* on rigid families (applied to the rigid family $\mathscr{S}_{\sigma,\tau}$) **[Hay14]**. As a result, we get:

LEMMA 2.19. *There is an order-isomorphism $\mathscr{C}(S \wedge T) \cong \mathscr{S}_{\sigma,\tau}$.*

This is reassuring, since we started from the interaction states to derive $S \wedge T$.

2.3.1. *Categorical picture.* This construction has an interesting algebraic characterization as a pullback in the category $\mathcal{E}$ of event structures:

PROPOSITION 2.20. *Let A be an event structure and $\sigma : S \to A$ and $\tau : T \to A$ be maps of event structures. There are maps of event structures $\Pi_1 : S \wedge T \to S$ and $\Pi_2 : S \wedge T \to T$ such that the following diagram is a pullback in the category $\mathcal{E}$:*



In other terms, for every maps $\alpha : X \to S$ and $\beta : X \to T$ such that $\sigma \circ \alpha = \tau \circ \beta$, there exists a unique map $\langle \alpha, \beta \rangle : X \to S \wedge T$ with $\Pi_1 \circ \langle \alpha, \beta \rangle = \alpha$ and $\Pi_2 \circ \langle \alpha, \beta \rangle = \beta$.

We write $\sigma \wedge \tau : S \wedge T \to A$ for either side of the commuting pullback square. This universal property makes some abstract properties (eg. associativity of interaction) straightforward to prove, and also gives confidence in the definition. We omit the proof (available in [**CCRW**]) since we prove a more general result later on (Lemma 2.47).

2.4. Non-closed interaction and composition. To get a category of strategies and model programming languages, we need to define a notion of pre-strategy from a game A to a game B. Following Joyal [**Joy77**], a pre-strategy from A to B will be a pre-strategy on the compound game $A^\perp \parr B$. The notation $\sigma : A$ to denote a pre-strategy on a game A is generalized to $\sigma : A \nrightarrow B$ to denote a pre-strategy from a game A to a game B. Copycat on A becomes a pre-strategy from A to itself – a candidate for an identity strategy.

How to compose a pre-strategy from A to B and a pre-strategy from B to C to get a pre-strategy from A to C? One plays on $A^\perp \parr B$, the other $B^\perp \parr C$, and the desired result on $A^\perp \parr C$, so it is not easily described as a closed interaction. To have them interact on B while the parts on A and C are left untouched, we build two pre-strategies on $A \parr B \parr C$ and take their interaction.

DEFINITION 2.21 (Open interaction). Let $\sigma : S \to A^\perp \parr B$ and $\tau : T \to B^\perp \parr C$ be pre-strategies. The maps $\sigma \parr C^\perp : S \parr C^\perp \to A^\perp \parr B \parr C^\perp$ and $A \parr T : A \parr T \to A \parr B^\perp \parr C$ have dual codomains. [β]

The **open interaction** of σ and τ is their interaction:

$$\tau \star \sigma = (\sigma \parr C^\perp) \wedge (A \parr \tau) : T \star S \to A \parr B \parr C.$$

In general $\tau \star \sigma$ is however not a pre-strategy on $A^\perp \parr C$ as one would like. To solve this issue, three solutions are carried out in the next sections:

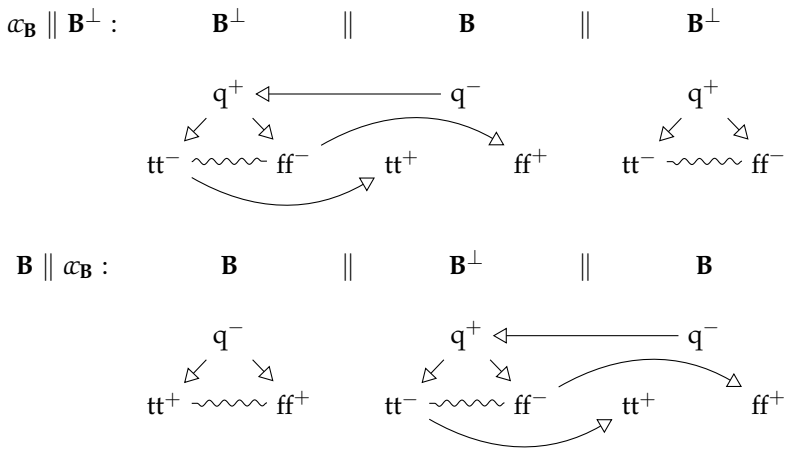
- Section 3 explores the possibility of *hiding* those events that are sent to B (called **internal events**) as done originally in [**RW11**]. Copycat becomes idempotent up to isomorphism, and we get a (bi)category of **strategies**, the pre-strategies that are invariant under composition with copycat.
- Hiding *all* internal events loses information up to weak bisimilarity. In Section 4, no hiding is performed, and pre-strategies (extended with internal events) are compared up to weak bisimulation so that to make copycat idempotent (modulo a minor restriction on games).

[β] At this point, it does not matter to find dual codomains as interaction is defined regardless. However, we point it out explicitly as in the setting of the next chapter, interaction will not be defined in such generality, and having dual codomains will become necessary.

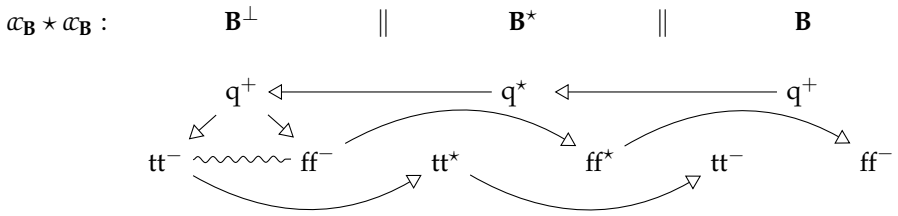
- To recover a category up to isomorphism, Section 5 restores some hiding: just enough to have an idempotent copycat (up to isomorphism) without losing any behaviour up to weak bisimilarity.

The following choices are summed up by the following example.

EXAMPLE 2.22. Consider the pre-strategy $\alpha_{\mathbf{B}} : \mathbb{C}_{\mathbf{B}} \to \mathbf{B}^{\perp} \parallel \mathbf{B}$, (or equivalently $\mathbf{B} \multimap \mathbf{B}$). We have $\alpha_{\mathbf{B}} \parallel \mathbf{B}$ and $\mathbf{B} \parallel \alpha_{\mathbf{B}}$ on $\mathbf{B} \parallel \mathbf{B} \parallel \mathbf{B}$:



Their interaction is:



For events on the left and on the right, since we want ultimately something that plays on $\mathbf{B}^{\perp} \parallel \mathbf{B}$, we know what polarities to put on them. However, events in the middle (that do not correspond to a move in the input or output game) are called *internal events* or *invisible events*. By convention, we say they have polarities $\star$, hence the notation $\mathbf{B}^*$.

The resulting event structure is not isomorphic to the event structure for copycat (there are more events). To solve this, Section 3 and 5 shrink the event structure to remove those synchronization events, while Section 4 modify the equivalence relation to identify it with $\alpha_{\mathbf{B}}$.

3. A category of total strategies

In this section, we omit the proofs and redirect the interested reader to [**CCRW**].

3.1. Composition via hiding. Hiding is performed through an operation called the **projection of events structures**. It removes some events deemed internal, and propagates causal dependencies and conflicts. The internal events are thought of as occurring in the background, any time after their visible dependencies occur.

DEFINITION 2.23 (Projection of event structures). Let E be an event structure and $V \subseteq E$ a subset of events (an event in V is called **visible**). The **projection** of E to V is the event structure $E \downarrow V$ defined as follows:

Events: V
Causality: $\leq_E \cap V^2$
Consistency: $X \in \mathrm{Con}_{E\downarrow V}$ if and only if $X \in \mathrm{Con}_E$ (and $X \subseteq V$)

Configurations of the resulting event structure are easily characterized:

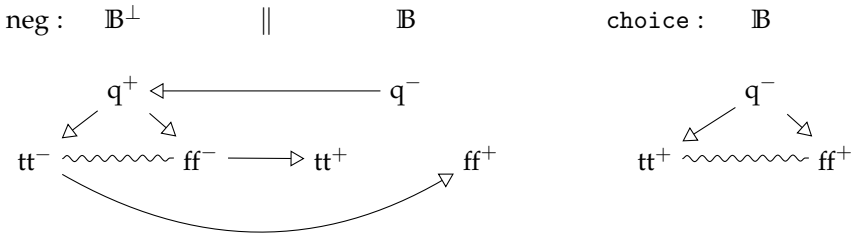
LEMMA 2.24. *Given an event structure E and $V \subseteq E$, configurations of $E \downarrow V$ are in one-to-one correspondence with configurations of E whose maximal events are in V.*

PROOF. The isomorphism maps a configuration $x \in \mathscr{C}(E)$ to $x \cap V \in E \downarrow V$ and a configuration $y \in \mathscr{C}(E \downarrow V)$ to $[y]_E$, the downclosure of y inside E. □

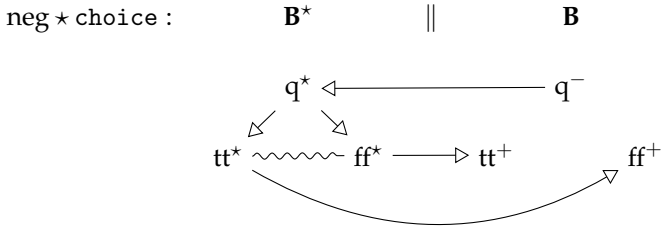
From this definition, composition of pre-strategies follows:

DEFINITION 2.25. Let $\sigma : S \to A^\perp \parallel B$ and $\tau : T \to B^\perp \parallel C$. Remember that their interaction is given by $\tau \star \sigma : T \star S \to A \parallel B \parallel C$. An event $e \in \tau \star \sigma$ is **visible** when $(\tau \star \sigma)(e) \notin B$ (which means that $(\tau \star \sigma)(e)$ is not in the B component of the disjoint union $A \parallel B \parallel C$). Writing V for the set of visible events, $T \odot S$ is defined as $(T \star S) \downarrow V$ and $\tau \odot \sigma : T \odot S \to A^\perp \parallel C$ as the restriction of $\tau \star \sigma$.

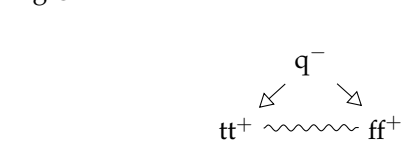
EXAMPLE 2.26 (Negation and nondeterministic choice). Remember the negation strategy $\mathrm{neg} : \mathbb{B} \nrightarrow \mathbb{B}$ and the nondeterministic boolean `choice`:



Their interaction gives (with $A = \emptyset$ – the empty game, $B = \mathbb{B}$ and $C = \mathbb{B}$):



After hiding, we get back the nondeterministic boolean as expected:

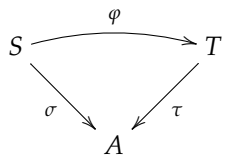


In what sense neg $\odot$ `choice` and `choice` are the same pre-strategy? They do not have *exactly* the same events (as events of neg $\odot$ `choice` are certain secured bijections), but their underlying event structures are *isomorphic*.

3.2. Isomorphism of strategies. The set S of events of a pre-strategy can be seen as names, which σ labels with moves from the game. The exact identity of those names does not matter, and composition heavily modifies these names. In particular $\mathfrak{a}_A \odot \sigma$ is never equal to σ because the names do not match.

As a result, the natural equality of pre-strategies is isomorphism:

DEFINITION 2.27 (Isomorphism of pre-strategies). Let $\sigma : S \to A$ and $\tau : T \to A$ be pre-strategies. An **isomorphism** between σ and τ is an isomorphism of event structures $\varphi : S \cong T$ (that is, an invertible map in $\mathcal{E}$) commuting with the action on the game given by σ and τ:



If two pre-strategies σ and τ are isomorphic, we write $\sigma \cong \tau$.

Composition is well-defined up to isomorphism:

LEMMA 2.28 (Isomorphism is a congruence). *For $\sigma, \sigma' : A \mathrel{-\!\!\vdash\!\!\succ} B$ and $\tau, \tau' : B \mathrel{-\!\!\vdash\!\!\succ} C$ such that $\sigma \cong \sigma'$ and $\tau \cong \tau'$, then $\tau' \odot \sigma' \cong \tau \odot \sigma$.*

Furthermore, composition is associative up to isomorphism:

LEMMA 2.29. *Composition of pre-strategies is associative up to isomorphism.*

3.3. A category of strategies. The natural candidate for an identity, copycat, is not an identity for many pre-strategies, as evidenced by the following examples:

EXAMPLE 2.30 (Failure of courtesy). Consider the game $A = \oplus_1 \; \oplus_2$ of two positive concurrent events, and the pre-strategy σ on A given by $\oplus_1 \multimap \oplus_2$.

The interaction $\mathfrak{a}_A \star \sigma$ is:

$$\begin{array}{ccc} A^\star & \| & A \\ & \mathfrak{a}_A & \\ \star_1 & \longrightarrow\!\!\triangleright & \oplus_1 \\ \sigma\downarrow\!\!\triangledown & \mathfrak{a}_A & \\ \star_2 & \longrightarrow\!\!\triangleright & \oplus_2 \end{array}$$

On this diagram, the causal links are annotated by the pre-strategy they come from. Since there is no path in the transitive closure between $\oplus_1$ and $\oplus_2$, the composition obtained by hiding A^* is reduced to the following:

$$\begin{array}{c} A \\ \\ \oplus_1 \\ \\ \oplus_2 \end{array}$$

which is not isomorphic to σ because of the missing causal link $\oplus_1 \rightarrowtriangle \oplus_2$.

Formally, this issue is due to the extra causal link between positive events that is absent from the game. It is however, not the only type of causal links that composition with copycat removes. Intuitively post-composition with copycat loses these causal links because of the asynchronous nature of our copycat which is unable to preserve most causal dependence. See Example 2.34 for more details.

Still, copycat is idempotent:

LEMMA 2.31. *For a game A, $\mathfrak{a}_A : A \vdash\!\!\succ A$ is idempotent, that is $\mathfrak{a}_A \odot \mathfrak{a}_A \cong \mathfrak{a}_A$.*

Using this fact, we can consider pre-strategies that are invariant under composition with copycat. A **strategy** on A is a pre-strategy σ on A such that there exists an isomorphism $\mathfrak{a}_A \odot \sigma \cong \sigma : A$. There are two ways to lift this notion to pre-strategies from A to B. The following lemma show that they coincide:

LEMMA 2.32. *Let $\sigma : A \vdash\!\!\succ B$ be a pre-strategy. The following are equivalent:*

(1) σ is a strategy on $A^\perp \parallel B$ (ie. $\mathfrak{a}_{A^\perp \parallel B} \odot \sigma \cong \sigma$),
(2) σ satisfies $\mathfrak{a}_B \odot \sigma \odot \mathfrak{a}_A \cong \sigma$.

PROOF. Consequence of Lemma 2.68 and Lemma 2.53, both to come. □

In that case, σ is called a **strategy** from A to B. Note that copycat in particular is a strategy since it satisfies (2) by idempotence. Assembling the pieces of the puzzle together, [**RW11**] finally deduces the following:

PROPOSITION 2.33. *The following data forms a category $\mathrm{CG}^{\cong}_{\odot}$:*

Objects: *games,*
Morphisms from A to B: *strategies from A to B up to isomorphism,*
Composition: *composition of strategies,*
Identity on A: *copycat strategy on A.*

The structure before quotient forms a *bicategory*, where 2-cells between $\sigma : S \to A$ and $\tau : T \to A$ are maps $f : S \to T$ making the obvious triangle commute. Details can be found in [**CCRW**], along with a proof that $\mathrm{CG}^{\cong}_{\odot}$ is compact-closed with $\parallel$ as tensor and $\cdot^\perp$ as dual. (See Section 6 for details on compact-closed categories).

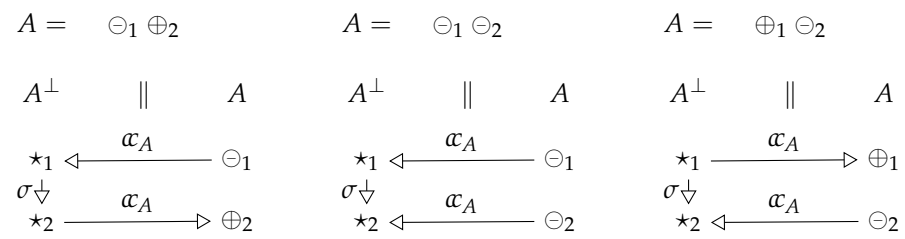
In this chapter, and the next one, we introduce a few categories of strategies that are variations on the composition and the notion of equality of strategies. To make this aspect clearer for the reader, their names are annotated: in superscript the equality between strategies, and in subscript the composition operation.

3.4. Concrete characterization of strategies. Our definition of strategy is very abstract; in particular among our examples which ones were strategies? Checking that a pre-strategy is a strategy involves studying its composition with copycat which can be quite tedious. The original article on concurrent games [**RW11**] provides a concrete and local characterization of strategies in terms of two conditions, *courtesy* (called innocence in [**RW11**]) and *receptivity*. Before introducing those conditions, we review the need for them in the following examples.

EXAMPLE 2.34 (The need for courtesy). As in Example 2.30, consider games of the form $A = e_1\ e_2$ with two concurrent moves e_1 and e_2. Depending on their polarities, there are four possible A. On this A, consider the strategy $\sigma = e_1 \rightarrowtriangle e_2$

(where σ is the identity on events). The four choices for A gives four choices of σ. Example 2.30 already studied the case of both e_1 and e_2 being positive.

We study the three remaining cases and depict $\mathfrak{a}_A \star \sigma$ for each:



After hiding, we observe that the only composition that is isomorphic to σ is the first one, when the extra causal link added by σ to the game is $\ominus \rightarrowtriangle \oplus$.

This prompts the following definition:

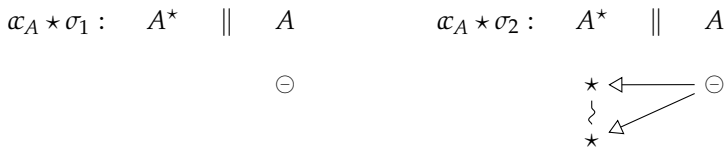
DEFINITION 2.35 (Courtesy). A pre-strategy $\sigma : S \to A$ is **courteous** whenever, if $s \rightarrowtriangle_S s'$ and, s is positive or s' is negative, then $\sigma s \rightarrowtriangle \sigma s'$.

Intuitively, copycat acts as an asynchronous forwarder. The only causal links that are stable under composition by this forwarder are the ones from negatives to positives, which corresponds to Player waiting for an Opponent move before playing. The causal links $\oplus \rightarrowtriangle \ominus$ correspond to forcing Opponent to wait for a Player move. The other two $\ominus \rightarrowtriangle \ominus$ and $\oplus \rightarrowtriangle \oplus$ represent controlling the order in which messages are exchanged in the network. Since our forwarder is asynchronous it cannot maintain this order – hence it disappears after composition.

Courtesy is however not enough to ensure that a pre-strategy is a strategy.

EXAMPLE 2.36 (Receptivity). Consider the game $A = \ominus$ and the two strategies $\sigma_1 = \emptyset$ (σ_1 never plays) and $\sigma_2 = \ominus \rightsquigarrow \ominus$ (σ_2 acknowledges the Opponent move in two nondeterministic ways.)

Their interaction with copycat gives:



which, after hiding, yield the pre-strategy $\ominus$ on A in both cases.

The problem here is either denying an Opponent move or performing a nondeterministic choice upon reception of an Opponent move. A strategy should treat reachable negative moves *linearly*: it cannot ignore them or duplicate them:

DEFINITION 2.37 (Receptivity). A pre-strategy $\sigma : S \to A$ is **receptive** when for all configuration $x \in \mathscr{C}(S)$ such that σx extends by a negative event $a^- \in A$, then there exists a unique extension $s \in S$ of x such that $\sigma s = a$.

Courtesy and receptivity exactly capture strategies:

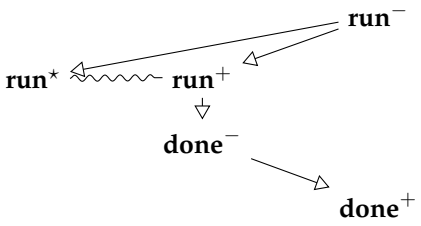
THEOREM 2.38. *Strategies coincide with courteous and receptive pre-strategies.*

4. Uncovered strategies up to weak bisimulation

In this section, we investigate another way of obtaining a category of strategies where no hiding is performed. Indeed, we can finally make formal the intuitions given in Chapter 1, that traditional hiding is too optimistic

EXAMPLE 2.39. Remember the strategy $\mathtt{sum}$ playing on $(\mathbf{proc} \parallel \mathbf{proc})^\perp \parallel \mathbf{proc}$ from Example 2.4. It can be seen as playing on $\mathbf{proc}^\perp \parallel (\mathbf{proc}^\perp \parallel \mathbf{proc})$. Computing the interaction $\mathtt{sum} \star \perp : \mathbf{proc}^\perp \parallel \mathbf{proc}$ where $\perp = \mathbf{run}^-$ is the minimal strategy on **proc** (representing divergence), yields:

$$\mathbf{proc}^\star \quad \parallel \quad (\mathbf{proc}^\perp \quad \parallel \quad \mathbf{proc})$$



Since $\perp$ never plays **done**, the left branch of sum disappears during the interaction. After hiding, it becomes apparent that $\mathtt{sum} \odot \perp \cong \alpha_{\mathbf{proc}}$. However, they do not have the same operational behaviour. Copycat on **proc** *always* runs its argument where $\mathtt{sum} \odot \perp$ might decide not to and interrogate $\perp$ instead, blocking the whole process. This is a phenomenon of *hidden divergence*. Because of that, the interpretation of nondeterministic languages in $\mathrm{CG}^{\cong}_{\odot}$ will only be adequate for *may-equivalence*. (Cf Chapter 4, Section 3 for a more detailed discussion)

To repair that, strategies should be allowed to have internal events that do not correspond to events of the game: the labelling function σ becomes *partial*.

DEFINITION 2.40 (Uncovered pre-strategies). An **uncovered pre-strategy** on a game A is a **partial map of event structures** $\sigma : S \rightharpoonup A$, *ie.* a partial function $S \rightharpoonup A$ satisfying the following familiar properties:

(1) $\sigma x \in \mathscr{C}(A)$ for $x \in \mathscr{C}(A)$,
(2) σ restricted to a consistent set is injective (*local injectivity*).

An uncovered pre-strategy from A to B is an uncovered pre-strategy on $A^\perp \parallel B$.

We reuse the notation $\sigma : A$ and $\sigma : A \mathrel{-\!\!\!\rightarrowtriangle} B$ from the previous section to introduce uncovered pre-strategies on a game. The (pre-)strategies of the previous section will be referred to as **covered (pre-)strategies** in the rest of the thesis.

Terminology on polarity. An event of an uncovered pre-strategy σ is **external** (or **visible**) if it is in the domain of σ, **internal** or **invisible** otherwise. The labelling σ does not equip S with a polarity function $S \to \{-,+\}$ since invisible events have no assigned polarities this way. In this case, σ equips S with a **partial polarity function** $S \to \{-,+,\star\}$ (where events undefined for σ have polarity $\star$), turning S into an **event structure with partial polarities**. Given such an event structure E, we define $E_\downarrow$ to be its projection to external events. Similarly given $x \in \mathscr{C}(E)$ we write $x_\downarrow$ for $x \cap E_\downarrow$ (it is a configuration of $E_\downarrow$) and $x_\star$ for the subset of internal events of x so that $x = x_\downarrow \uplus x_\star$.

An event is **nonnegative** when it is positive or internal: such events correspond to Player actions. The notations $\subseteq^+$ and $\subseteq^-$ are extended to $\subseteq^\star$: $x \subseteq^\star y$ when y is an extension of x by only invisible events. Given an uncovered pre-strategy $\sigma : S \rightharpoonup A$ we write $\sigma_\downarrow : S_\downarrow \to A$ for the corresponding covered strategy.

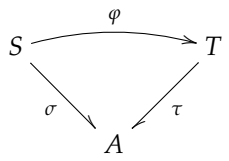
Note that Lemma 2.41 generalizes easily to partial maps:

LEMMA 2.41. *Let $f, g : A \rightharpoonup B$ be partial maps of event structures such that for all configuration $x \in \mathscr{C}(A)$ we have $fx = gx$. Then $f = g$.*

PROOF. Similar proof as in the total case, but here $f[a] \setminus f[a)$ is either empty or a singleton depending on whether f is defined at $a \in A$. □

Isomorphism naturally generalizes to uncovered pre-strategies:

DEFINITION 2.42. Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be two uncovered pre-strategies. An isomorphism between σ and τ is an isomorphism $\varphi : S \cong T$ such that the following triangle commutes (as partial maps):



4.1. Interaction of uncovered strategies. In this setting, composition simply becomes interaction since we do not want to hide anything. Since pre-strategies are now partial maps, the definition of the interaction needs to be updated.

4.1.1. *Closed interaction.* First, we define the *closed* interaction of uncovered pre-strategies. As before, polarities do not matter and we consider simply partial maps of event structures. Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be partial maps of event structures. To define the new interaction states, we need to slightly change the definition of Section 2.3.

A pair $(x, y) \in \mathscr{C}(S) \times \mathscr{C}(T)$, such that $\sigma_\downarrow x = \tau_\downarrow y \in \mathscr{C}(A)$, induces a bijection $\varphi_{x,y} : x \parallel y_\star \simeq x_\star \parallel y$ defined by local injectivity of σ and τ:

$$\begin{aligned} \varphi_{x,y}(0,s) &= (0,s) && (s \in x_\star) \\ \varphi_{x,y}(0,s) &= (1, \tau^{-1}(\sigma s)) && (s \in x_\downarrow) \\ \varphi_{x,y}(1,t) &= (1,t) \end{aligned}$$

Viewing $y_\star$ and $x_\star$ as discrete orders (the ordering relation is the equality), $\varphi_{x,y}$ is a bijection between partial orders. An **interaction state** of σ and τ is a pair $(x, y) \in \mathscr{C}(S) \times \mathscr{C}(T)$ with $\sigma_\downarrow x = \tau_\downarrow y$ such that the bijection $\varphi_{x,y}$ is secured. As a result $\varphi_{x,y}$ is naturally partial ordered.

As for covered strategies, write $\mathscr{S}_{\sigma,\tau}$ for the set of interaction states of σ and τ. This terminology is justified as in the case of covered strategies, both notions of interaction states coincide. We can now apply the same trick of Definition 2.18:

DEFINITION 2.43 (Closed interaction of uncovered pre-strategies). Let A be an event structure, and $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be partial maps of event structures. The following data defines an event structure $S \wedge T$:

- (*Events*) Those interaction states (x, y) such that $\varphi_{x,y}$ has a top element.
- (*Causality*) Inclusion of graphs.

- (*Consistency*) A finite set of interaction state $X \subseteq S \wedge T$ is consistent iff its pairwise union $\bigcup X$ is an interaction state in $\mathscr{S}_{\sigma,\tau}$.

Projections become partial maps $\Pi_1 : S \wedge T \rightharpoonup S$ and $\Pi_2 : S \wedge T \rightharpoonup T$ defined as follows. For $(x, y) \in \mathscr{S}_{\sigma,\tau}$ $\Pi_1(x, y)$ is defined to $s \in S$ whenever the top-element of $\varphi_{x,y}$ is $((1, s), w_2)$ for some $w_2 \in x_\star \parallel y$. The map Π_2 is defined similarly.

We write $\sigma \wedge \tau$ for $\sigma \circ \Pi_1 = \tau \circ \Pi_2 : S \wedge T \rightharpoonup A$. As a result, Π_1 is undefined only on events of $S \wedge T$ corresponding to internal events of T and similarly Π_2 is undefined on events corresponding to neutral events of S.

4.1.2. *Properties of closed interaction.* When proving properties about the interaction of pre-strategies, a concrete understanding of the interaction will be handy. Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be partial maps.

As desired, configurations of $S \wedge T$ indeed correspond to interaction states:

LEMMA 2.44. *The map $z \mapsto (\Pi_1 z, \Pi_2 z) : \mathscr{C}(S \wedge T) \to \mathscr{S}_{\sigma,\tau}$ is an order-isomorphism. Moreover, given $z \in \mathscr{C}(S \wedge T)$, there is an order-isomorphism $\mathrm{top}_z : z \cong \varphi_{\Pi_1 z, \Pi_2 z}$.*

PROOF. The proof follows from general considerations on rigid families [**Hay14**]. For the sake of completeness, we provide an elementary proof.

Write $\Psi : \mathscr{C}(S \wedge T) \to \mathscr{S}_{\sigma,\tau}$ defined by $\Psi(z) = (\Pi_1 z, \Pi_2 z)$. It is well-defined because the graph of $\varphi_{\Pi_1 z, \Pi_2 z}$ is exactly

$$\bigcup_{(x,y) \in z} \varphi_{x,y}.$$

(Remember that, formally, z is a set of interaction states.)

Define $\Psi^{-1} : \mathscr{S}_{\sigma,\tau} \to \mathscr{C}(S \wedge T)$ as follows:

$$\Psi^{-1}(x, y) = \{(x', y') \in S \wedge T \mid x' \subseteq x \;\&\; y' \subseteq y\}.$$

The equations $\Psi \circ \Psi^{-1} = \mathrm{id}_{\mathscr{S}_{\sigma,\tau}}$ and $\Psi^{-1} \circ \Psi = \mathrm{id}_{\mathscr{C}(S \wedge T)}$ are simple calculations.

Finally, consider $z \in \mathscr{C}(S \wedge T)$. Elements z_0 of z are prime interaction states, as such have a top element $\mathrm{top}_z(z_0)$ in $\varphi_{\Pi_1 z, \Pi_2 z}$. This defines a monotonic map $z_0 \mapsto \mathrm{top}_z(z_0) : z \to \varphi_{\Pi_1 z, \Pi_2 z}$. Conversely, given $p \in \varphi_{\Pi_1 z, \Pi_2 z}$, the set $\{p' \in \varphi_{\Pi_1 z, \Pi_2 z} \mid p' \leq p\}$ is the graph of a secured bijection $\varphi_{x,y}$. The mapping $p \mapsto (x, y)$ defines the inverse $\varphi_{\Pi_1 z, \Pi_2 z} \to z$. □

LEMMA 2.45. *We have the following properties:*

1. *If $e \twoheadrightarrow e'$ in $S \wedge T$, then $\Pi_1 e \twoheadrightarrow_S \Pi_1 e'$ (with both defined) or $\Pi_2 e \twoheadrightarrow_T \Pi_2 e'$ (with both defined).*
2. *Assume we have $x \in \mathscr{C}(S \wedge T)$ with two incompatible extensions e and e'. Then $\Pi_1 e$ and $\Pi_1 e'$ are defined and are incompatible extensions of $\Pi_1 x$, or $\Pi_2 e$ are defined and $\Pi_2 e'$ are incompatible extensions of $\Pi_2 x$.*

PROOF. The proof exploits the definition of events of $S \wedge T$ as prime interaction states. Remember that an event $e \in S \wedge T$ *is* an interaction state inducing a secured bijection $\varphi_{\Pi_1[e], \Pi_2[e]} : \Pi_1[e] \parallel (\Pi_2[e])_\star \simeq (\Pi_1[e])_\star \parallel \Pi_2[e]$.

Write N_σ and N_τ for the set of internal events of σ and τ respectively, viewed as event structures where causality is equality and all finite sets are consistent. The partial projections can be turned into total maps:

$$\overline{\Pi_1} : S \wedge T \to S \parallel N_\tau \qquad \overline{\Pi_2} : S \wedge T \to N_\sigma \parallel T.$$

For any $e_0 < e$, then $(\overline{\Pi_1} e_0, \overline{\Pi_2} e_0) \in \varphi_{\Pi_1[e], \Pi_2[e]}$.

(1) From $e \rightarrowtriangle e'$, we deduce that $(\overline{\Pi_1}e, \overline{\Pi_2}e) \rightarrowtriangle_{\varphi_{\Pi_1 e', \Pi_2 e'}} (\overline{\Pi_1}e', \overline{\Pi_2}e')$. By definition of the induced order on an interaction state, this implies that either $\overline{\Pi_1}e \rightarrowtriangle_{S\|N_\tau} \overline{\Pi_1}e'$ or $\overline{\Pi_2}e \rightarrowtriangle_{N_\sigma\|T} \overline{\Pi_2}e'$. Since N_σ and N_τ are discrete partial order, this is equivalent to $\overline{\Pi_1}e \rightarrowtriangle_S \overline{\Pi_1}e'$ or $\overline{\Pi_2}e \rightarrowtriangle_T \overline{\Pi_2}e'$, as desired.

(2). Write $X = x \cup \{e, e'\}$, and φ for the secured bijection $\varphi_{\Pi_1 x, \Pi_2 x}$ (well-defined by Lemma 2.44. Assume both $\Pi_1 X$ and $\Pi_2 X$ are configurations of S and T respectively. Since the projections of e and e' are concurrent both in S and T, the bijection $\varphi \cup \{(\overline{\Pi_1}e, \overline{\Pi_2}e), (\overline{\Pi_1}e', \overline{\Pi_2}e')\}$ is secured, therefore X' is a configuration of $S \wedge T$ by Lemma 2.44, which is absurd. □

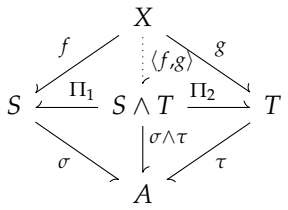
4.1.3. *Universal property of the interaction.* Previously, we have seen that synchronization $\sigma \wedge \tau$ of total maps of event structures is actually a pullback; and as a result enjoys a universal property. This universal property is neat as it allows abstract reasoning on interactions and provides a justification for the definition. However, even though the category of event structures and *partial* maps does have pullbacks, $\sigma \wedge \tau$ is not a pullback of the partial maps σ and τ.

EXAMPLE 2.46. Let A be the empty game and σ and τ be the empty partial map $S = T = \{\star\} \rightharpoonup A$. A calculation gives that $S \wedge T$ contains two events $\star_\sigma$ and $\star_\tau$ such that $\Pi_1(\star_\sigma)$ is defined but not $\Pi_2(\star_\sigma)$ and vice-versa for $\star_\tau$.

However, the pullback of σ and τ inside the category of partial maps has a third event $\star_{\sigma,\tau}$ on which **both** Π_1 and Π_2 are defined.

This is due to pullbacks of partial maps synchronizing on the internal events. In our construction, this does not happen: neutral events of σ and τ live in different worlds. This intuition is formally expressed as the following property on our projections Π_1 and Π_2: if $\Pi_1 p$ and $\Pi_2 p$ are both defined, for an event $p \in S \wedge T$, then p is visible, ie. $\sigma(\Pi_1 p) = \tau(\Pi_2 p)$ is defined. We can prove that our triple $(S \wedge T, \Pi_1, \Pi_2)$ is universal among triples having this property:

LEMMA 2.47. *Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be partial maps of event structures. Let $(X, f : X \rightharpoonup S, g : X \rightharpoonup T)$ be a triple such that the following outer square commutes:*



Assume that for all $p \in X$ with $f\ p$ and $g\ p$ defined, $\sigma(f\ p) = \tau(g\ p)$ is defined. Then, there exists a unique map $\langle f, g\rangle : X \to S \wedge T$ making the two upper triangles commute.

The action of $\langle f, g\rangle$ is more easily described on configurations than on events. Fortunately, it is possible recover the action on events, modulo some assumptions:

LEMMA 2.48 (Mapification). *Let A, B be event structures. Let $p : \mathscr{C}(A) \to \mathscr{C}(B)$ be a monotonic map between the configuration domains. If p satisfies both:*

(0) $p\,\emptyset = \emptyset$,

(1) for $x, y \in \mathscr{C}(A)$ such that $x \cup y \in \mathscr{C}(A)$, then

$$p(x \cup y) \setminus px = (py \setminus p(x \cap y)),$$

(2) p is non-inflating: if $x \mathrel{-\!\!\subset} y$ then $|py| \leq |px| + 1$ ($|\cdot|$ denoting cardinality).

then, there exists a (necessarily unique) partial map of event structures $f : A \rightharpoonup B$ with $fx = px$ for all $x \in \mathscr{C}(A)$. Moreover, f is total if and only if (2) is always an equality.

PROOF. Remark that, by (2), $p[a] \setminus p[a)$ is either empty or a singleton. Define the partial function $f : A \rightharpoonup B$ as follows: it is defined on $a \in A$ to $b \in B$ if and only $p[a] \setminus p[a) = \{b\}$.

First, by induction on $x \in \mathscr{C}(A)$ we show that $fx = px$. For $x = \emptyset$, it is given by assumption (0). Assume that $x \in \mathscr{C}(A)$ such that $fx = px$ extends by $a \in A$ to x'. We have (taking $x = x$ and $y = [a]$ in (2)):

$$p(x') = p(x \cup [a]) = p(x) \cup (p([a] \setminus p[a)).$$

If f is defined at a, then $p[a] \setminus p[a) = \{fa\}$, if f is undefined at a, then $p[a] \setminus p[a) = \emptyset$. In both cases, we have $fx' = px'$.

We now check f is a map of event structures. If $x \in \mathscr{C}(A)$, then $fx = px \in \mathscr{C}(B)$. For local injectivity, assume distinct $a, a' \in x \in \mathscr{C}(A)$ such that fa and fa' are defined and equal to b. Assume that, (a, a') is a minimal pair satisfying this property, so that $b \notin f[a] \cap f[a']$. By (2), it follows that:

$$f([a] \cup [a']) \setminus f[a] = f[a'] \setminus (f[a] \cap f[a']).$$

However, b belongs to the right-hand term, but not to the left-hand term: absurd.

Finally, f is total if and only if $x \overset{a}{\mathrel{-\!\!\subset}} y$, $p(y) = \{b\} \cup p(x)$ for some $b \in B$. □

Using this construction, we can now prove the universal property:

PROOF. (Of Lemma 2.47)

Uniqueness. Assume we have $\iota_1, \iota_2 : X \to S \wedge T$ making the two diagrams commute. For a configuration $x \in \mathscr{C}(X)$, a calculation yields:

$$(\Pi_1(\iota_1 x), \Pi_2(\iota_1 x)) = (fx, gx) = (\Pi_1(\iota_2 x), \Pi_2(\iota_2 x))$$

which implies $\iota_1 = \iota_2$ by Lemmata 2.44 and Lemma 2.41.

Existence. Write $\Psi : \mathscr{S}_{\sigma,\tau} \cong \mathscr{C}(S \wedge T)$ for the isomorphism given by Lemma 2.44. We use mapification (Lemma 2.48) to build $\langle f, g \rangle$.

If $x \in \mathscr{C}(X)$ then $(fx, gx) \in \mathscr{S}_{\sigma,\tau}$: by Lemma 2.12, any cycle in the secured bijection $(fx) \parallel (gx)_\star \simeq (fx)_\star \parallel gx$ can be traced back by to a cycle in x which is absurd. Hence, the mapping $x \in \mathscr{C}(X) \mapsto \Psi(fx, gx) \in \mathscr{C}(S \wedge T)$ is a well-defined monotonic map $\langle f, g \rangle$, and, at the level of configurations, these hold:

$$\Pi_1 \circ \langle f, g \rangle = f \quad \text{and} \quad \Pi_2 \circ \langle f, g \rangle = g.$$

We now check the mapification conditions holds for $\langle f, g \rangle$. First, since Ψ is an order-isomorphism, (1) is a simple calculation. For (2), any $x \in \mathscr{C}(X)$ can be decomposed in visible events, neutral events of f, neutral events for g as follows:

$$x_v = x \cap \mathrm{dom}(\sigma \circ f) \qquad x_f = (x \cap \mathrm{dom}\, f) \setminus x_v \qquad x_g = (x \cap \mathrm{dom}\, g) \setminus x_v.$$

Because f and g do not synchronize on neutral events, these three sets are disjoint. However they do not form a partition, as there could be event in x on which neither f nor g are defined. In any case, we have the desired inequality:

$$|\langle f, g\rangle x| = |\varphi_{fx,gx}| \leq |fx| + |(gx)_\star| = (|x_v| + |x_f|) + |x_g| \leq |x|$$

as the union is disjoint. As a result, by mapification $\langle f, g\rangle$ induces a map, that we also write $\langle f, g\rangle : X \rightharpoonup S \wedge T$, satisfying the desired equations by Lemma 2.41. □

One consequence of this universal property is an easy proof of associativity:

LEMMA 2.49. *For $\sigma : S \rightharpoonup A, \tau : T \rightharpoonup A, v : U \to A$, $(\sigma \wedge \tau) \wedge v \cong \sigma \wedge (\tau \wedge v)$.*

PROOF. The maps $\Pi_2 \circ \Pi_1 : (S \wedge T) \wedge U \rightharpoonup T$ and $\Pi_2 : (S \wedge T) \wedge U \rightharpoonup U$ do not synchronize on neutral events, as a result, there exists a map $\varphi_1 = \langle \Pi_2 \circ \Pi_1, \Pi_2\rangle : (S \wedge T) \wedge U \rightharpoonup (T \wedge U)$. Since φ_1 and $\Pi_1 \circ \Pi_1 : (S \wedge T) \wedge U \rightharpoonup S$ do not synchronize on neutral events either, then $\varphi_2 = \langle \Pi_1 \circ \Pi_1, \varphi_1\rangle : (S \wedge T) \wedge U \rightharpoonup S \wedge (T \wedge U)$ is well-defined. Similarly, we can build a map in the other direction, which is an inverse to φ_2 by Lemma 2.41. □

We note in passing that mapification has an interesting consequence: its assumptions are satisfied when p is an order-isomorphism:

LEMMA 2.50. *Let A and B be event structures and let $\varphi : \mathscr{C}(A) \cong \mathscr{C}(B)$ be an order-isomorphism. There is a (necessarily unique by Lemma 2.41) isomorphism of event structures $\hat{\varphi} : A \cong B$ such that for all $x \in \mathscr{C}(A)$, $\hat{\varphi}x = \varphi x$.*

PROOF. Direct consequence of Lemma 2.48. □

This result along with Lemma 2.41 allows us to prove isomorphisms of strategies by reasoning only at the level of configurations.

4.1.4. *Open interaction.* To deduce the open interaction, we simply proceed as in previous section. Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$. As before, we consider $(\sigma \parallel C) \wedge (A \parallel \tau) : T \star S \rightharpoonup A \parallel B \parallel C$. Write $\mathfrak{h}_B^{A\parallel B\parallel C}$ for the partial map $A \parallel B \parallel C \rightharpoonup A \parallel C$. To get the interaction, $\tau \star \sigma$ we simply post-compose $(\sigma \parallel C) \wedge (A \parallel \sigma)$ by $\mathfrak{h}_B^{A\parallel B\parallel C}$, marking the events sent to B as internal:

$$\tau \circledast \sigma = \mathfrak{h}_B^{A\parallel B\parallel C} \circ ((\sigma \parallel C) \wedge (A \parallel \sigma)) : T \star S \rightharpoonup A^\perp \parallel C$$

We use a different notation ($\circledast$ instead of $\star$) to emphasize the fact that we have changed the codomain. However, the underlying event structure is the same, and we write sometimes $T \circledast S = T \star S$ for uniformity purposes. To prove associativity of interaction, we need laws governing the distributivity between this operation and parallel composition and $\wedge$:

LEMMA 2.51. *For pre-strategies $\sigma : S \rightharpoonup A \parallel B \parallel C$ and $\tau : T \rightharpoonup C$:*

$$\mathfrak{h}_B^{A\parallel B\parallel C} \circ (\sigma \wedge (A \parallel B \parallel \tau)) \cong (\mathfrak{h}_B^{A\parallel B\parallel C} \circ \sigma) \wedge (A \parallel \tau)$$

PROOF. Write U_1 for the event structure corresponding to the left-hand term and U_2 for that of the right-hand term. Both satisfy the universal property of Lemma 2.47. In particular we have projections:

$$\Pi_1^{U_1} : U_1 \rightharpoonup S \qquad \Pi_2^{U_1} : U_1 \rightharpoonup A \parallel B \parallel T$$
$$\Pi_1^{U_2} : U_2 \rightharpoonup S \qquad \Pi_2^{U_2} : U_2 \rightharpoonup A \parallel T$$

As a result, by the universal property we have a map $U_1 \rightharpoonup U_2$ induced by $\Pi_1^{U_1} : U_1 \rightharpoonup S$ and $\mathfrak{h}_B \circ \Pi_2^{U_1} : U_2 \rightharpoonup A \parallel B \parallel T \rightharpoonup A \parallel T$. The other way around is slightly more subtle. In U_2, σ and $B \parallel \tau$ do not synchronize on B but in U_1 they do. However, since $B \parallel \tau$ plays as the identity on B, this synchronization on B does not constrain σ. Formally, there exists a map:

$$\psi : U_2 \to A \parallel B \parallel T$$

such that $\Pi_1^{U_2}$ together with ψ induce the desired inverse $U_2 \to U_1$.

The map ψ is defined as follows:

$$\psi(p) \in A \parallel B \parallel T = \begin{cases} \sigma(\Pi_1(p)) & \text{if } \sigma(\Pi_1(p)) \text{ is defined and in } A \parallel B \\ \Pi_2 p & \text{otherwise – equivalently, } \Pi_2 p \in T \end{cases}$$

This workaround is necessary to ensure that the image in B coincides with σ. □

From this lemma and associativity of $\wedge$, we can deduce associativity of $\circledast$:

LEMMA 2.52. *For σ, τ, υ composable uncovered pre-strategies, we have:*

$$\sigma \circledast (\tau \circledast \upsilon) \cong (\sigma \circledast \tau) \circledast \upsilon.$$

PROOF. Let $\sigma : S \rightharpoonup A^\perp \parallel B, \tau : T \rightharpoonup B^\perp \parallel C$ and $\upsilon : U \rightharpoonup C^\perp \parallel D$ be uncovered pre-strategies. Using Lemma 2.49, we have:

$$\begin{aligned}
(\sigma \circledast \tau) \circledast \upsilon &\cong \mathfrak{h}_C^{A\parallel C\parallel D} \circ (((\mathfrak{h}_B^{A\parallel B\parallel C} \circ ((\sigma \parallel C) \wedge (A \parallel \tau))) \parallel D) \wedge (A \parallel \upsilon)) \\
&\qquad \left\{ \quad (\mathfrak{h}_B^{A\parallel B\parallel C} \circ \sigma) \parallel D \cong \mathfrak{h}_B^{A\parallel B\parallel C\parallel D} \circ (\sigma \parallel D) \quad \right\} \\
&\cong \mathfrak{h}_C^{A\parallel C\parallel D} \circ ((\mathfrak{h}_B^{A\parallel B\parallel C\parallel D} \circ ((\sigma \parallel C \parallel D) \wedge (A \parallel \tau \parallel D))) \wedge (A \parallel \upsilon)) \\
&\qquad \{ \quad \text{Lemma 2.51} \quad \} \\
&\cong \mathfrak{h}_C^{A\parallel C\parallel D} \circ \mathfrak{h}_B^{A\parallel B\parallel C\parallel D}((\sigma \parallel C \parallel D) \wedge (A \parallel \tau \parallel D) \wedge (A \parallel B \parallel \upsilon)) \\
&\qquad \{ \quad A \parallel (\sigma \wedge \tau) \cong (A \parallel \sigma) \wedge (A \parallel \tau) \quad \} \\
&\cong \mathfrak{h}_C^{A\parallel C\parallel D} \circ \mathfrak{h}_B^{A\parallel B\parallel C\parallel D}((\sigma \parallel C \parallel D) \wedge (A \parallel ((\tau \parallel D) \wedge (B \parallel \upsilon))) \\
&\cong \mathfrak{h}_B^{A\parallel B\parallel D} \circ \mathfrak{h}_C^{A\parallel B\parallel C\parallel D}((\sigma \parallel C \parallel D) \wedge (A \parallel ((\tau \parallel D) \wedge (B \parallel \upsilon))) \\
&\cong \sigma \circledast (\tau \circledast \upsilon). \quad \square
\end{aligned}$$

Finally, composition of uncovered pre-strategies commutes with hiding:

LEMMA 2.53. *For $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ uncovered pre-strategies,*

$$(\tau \circledast \sigma)_\downarrow \cong \tau_\downarrow \odot \sigma_\downarrow$$

PROOF. The proof is delayed until Section 6. □

4.2. Weak bisimulation. As discussed in Example 2.22, composition with copycat adds extra synchronization events. To identify uncovered pre-strategies up to those synchronization events, we use the standard notion of *weak bisimulation* on the labeled transition system generated by the corresponding event structures:

DEFINITION 2.54 (Weak bisimulation). Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A$ be two uncovered pre-strategies. A weak bisimulation between σ and τ is a set $\mathcal{R}$ of pairs of the form (x, y), where $x \in \mathscr{C}(S)$, $y \in \mathscr{C}(T)$ satisfying the following axioms:

(1) $(\emptyset, \emptyset) \in \mathcal{R}$

(2) If $(x, y) \in \mathcal{R}$ with $x \stackrel{s}{\longrightarrow\!\!\subset}$ and s is visible, then there exists $y \subseteq^\star y'$ such that $y' \stackrel{t}{\longrightarrow\!\!\subset}$ and $(x \cup \{s\}, y' \cup \{t\}) \in \mathcal{R}$.
(And the symmetric condition for a visible extension of y)

(3) If $(x, y) \in \mathcal{R}$ with $x \stackrel{s}{\longrightarrow\!\!\subset}$ and s invisible then there exists $y \subseteq^\star y'$ and $(x \cup \{s\}, y') \in \mathcal{R}$.
(And the symmetric condition for a invisible extension of y)

Two uncovered pre-strategies are **weakly bisimilar** when there exists a weak bisimulation $\mathcal{R}$ between them. We write $\sigma \approx \tau$.

EXAMPLE 2.55. Our example 2.39 can be used to illustrate weak bisimulation by showing that $\texttt{sum} \circledast \perp$ and $\mathfrak{a}_{\mathbf{proc}}$ are not weakly bisimilar. Indeed, we must have $(\{\mathbf{run}^-\}, \{\mathbf{run}^-\})$ in any bisimulation between them. From there $\texttt{sum} \circledast \perp$ can do an internal transition, performing the left (internal) $\mathbf{run}^*$ whereas $\mathfrak{a}_{\mathbf{proc}}$ has no choice but to stand still. From then on, $\texttt{sum} \circledast \perp$ is stuck whereas $\mathfrak{a}_{\mathbf{proc}}$ can still perform visible transitions.

LEMMA 2.56. *Weak bisimulation is a congruence (for $\circledast$).*

PROOF. Let $\sigma : S \rightharpoonup A^\perp \parallel B$, $\sigma' : S' \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$, $\tau' : T' \rightharpoonup B^\perp \parallel C$. Assume $\mathcal{R}$ is a bisimulation between σ and σ', and $\mathcal{R}'$ between τ and τ'. Define $\mathcal{R} \parallel C$ as the relation:

$$\{(x \parallel y, x' \parallel y) \mid x\mathcal{R}x', y \in \mathscr{C}(C)\}$$

and similarly $A \parallel \mathcal{R}'$. They are bisimulations $\sigma \parallel C \approx \sigma' \parallel C$ and $A \parallel \tau \approx A \parallel \tau'$ respectively. Define $\mathcal{R}' \circledast \mathcal{R}$ as follows:

$$x(\mathcal{R}' \circledast \mathcal{R})y \quad \text{iff} \quad \Pi_2 x(A \parallel \mathcal{R}')\Pi_2 y \text{ and } \Pi_1 x(\mathcal{R} \parallel C)\Pi_1 y.$$

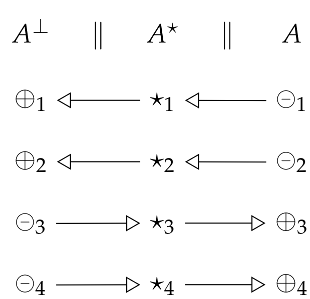
Checking this is a bisimulation is routine. □

To continue, we need to study which uncovered pre-strategies are invariant under copycat. However, unlike in Section 3, copycat is not always idempotent:

EXAMPLE 2.57. Let $A = \ominus_1 \ominus_2 \oplus_3 \oplus_4$ with trivial causality and consistency:

$$X \in \mathrm{Con}_A \text{ iff } |X| \geq 3 \Rightarrow (\ominus_2 \notin X \vee \oplus_3 \notin X).$$

In a context where two events already occurred, $\ominus_2$ and $\oplus_3$ become mutually exclusive. In particular $\{\ominus_2, \oplus_3\}$ is a maximal configuration of A. The interaction of copycat with itself gives the following uncovered pre-strategy:



The conflict, as it is not binary, is not represented on the above picture. Any bisimulation between $\mathfrak{a}_A \circledast \mathfrak{a}_A$ and $\mathfrak{a}_A$ must relate the minimal configurations of $\mathfrak{a}_A \circledast \mathfrak{a}_A$ and $\mathfrak{a}_A$, one of which is $\{\ominus_1, \ominus_2, \ominus_3, \ominus_4\} \in \mathscr{C}(A^\perp \parallel A)$. From there, $\mathfrak{a}_A \circledast \mathfrak{a}_A$ can do a silent transition to $\{\ominus_1, \ominus_2, \ominus_3, \ominus_4, \star_2, \star_3\}$ (and $\mathfrak{a}_A$ does nothing since there are no internal events in $\mathfrak{a}_A$). But this configuration of $\mathfrak{a}_A \circledast \mathfrak{a}_A$ is maximal (in particular because the configuration in the middle in maximal is A) whereas the corresponding configuration of $\mathfrak{a}_A$ is not: hence they cannot be bisimilar.

To avoid this problem, from now on, we only consider race-free games, *ie.* games that do not exhibit mutual exclusion between positive and negative events:

DEFINITION 2.58 (Race-freeness). An event structure with polarities A is **race-free** when for all incompatible extensions a_1 and a_2 of a configuration of A, then a_1 and a_2 must have the same polarity.

The key consequence of race-freeness is as follows:

LEMMA 2.59. *Let A be a race-free event structure with polarities. For all configuration $x \parallel y \in \mathscr{C}(\mathbb{C}_A)$, then $x \cup y \in \mathscr{C}(A)$.*

PROOF. By Lemma 2.11, $y \supseteq^- x \cap y \subseteq^+ x$. If $x \cup y \notin \mathscr{C}(\mathbb{C}_A)$, this means that there exists $x \cap y \subseteq z \subseteq x \cup y$ with two incompatible extensions $a_1 \in x$ and $a_2 \in y$. Since, a_1 is positive and a_2 negative which contradicts race-freeness. □

Race-freeness is preserved by all our constructions and all the games we consider in this thesis are race-free, so for our purposes, this is not a very restrictive constraint. That copycat on race free games is idempotent is a consequence of our Proposition 2.67 to come.

It is to be noted though, that race-freeness is not necessary to guarantee idempotence of copycat, and a notion of *transitive game* should be sufficient: those where the relation "$x \triangleleft y$ iff $x \parallel y$ is a $+$-maximal configuration of $\mathbb{C}_A$" is a transitive relation between configurations of A (see **[CHLW14]** for more details).

4.3. A compact-closed category. Restricting to race-free games, we now investigate conditions on pre-strategies for them to be invariant under copycat. We first remark that generalizing courtesy and receptivity is not enough. Indeed, new behaviours of uncovered pre-strategies are not invariant under copycat.

EXAMPLE 2.60. Consider the game $A = \mathrm{tt}^+\ \mathrm{ff}^+$ and $\sigma = \mathrm{tt}^+ \rightsquigarrow \mathrm{ff}^+$. Its composition, as an uncovered pre-strategy, with copycat is:

$$\begin{array}{ccc} A^\star & \parallel & A \\ \star_{\mathrm{tt}} & \longrightarrow\!\triangleright & \mathrm{tt}^+ \\ \wr & & \\ \star_{\mathrm{ff}} & \longrightarrow\!\triangleright & \mathrm{ff}^+ \end{array}$$

It is not weakly bisimilar to σ as $\mathfrak{a}_A \circledast \sigma$ can do an internal transition to choose the result of the computation, and disable the other one, while σ has to stand still (there is no internal event) and can still play both moves afterward.

This motivates the definition of *uncovered strategy*:

DEFINITION 2.61 (Uncovered strategy). A **uncovered strategy** is an uncovered pre-strategy $\sigma : S \nrightarrow A$ satisfying furthermore the following conditions:

Courtesy: For $s \rightarrow_S s'$ and $\mathrm{pol}(s) = +$ or $\mathrm{pol}(s') = -$ then $\sigma s \rightarrow \sigma s'$,
Receptivity: For $x \in \mathscr{C}(S)$ such that σx extends by a negative event a then there exists a unique extension $s^- \in S$ of x with $\sigma s = a$.
Secrecy: For $x \in \mathscr{C}(S)$ with two incompatible extensions by s_1 and by s_2, then s_1 and s_2 are either both internal events or both negative events.

Secrecy forces nondeterministic actions of a strategy to be internal. The first two axioms are straightforward generalizations of courtesy and receptivity. As a result, the hiding of an uncovered strategy is a covered strategy:

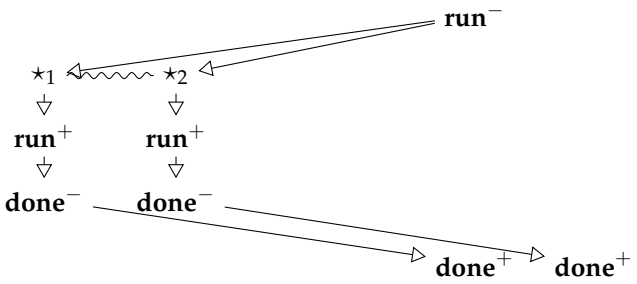
LEMMA 2.62. *For $\sigma : S \rightharpoonup A$ an uncovered strategy, $\sigma_\downarrow$ is a covered strategy.*

PROOF. *Courtesy.* Assume we have $s \rightarrow_{S_\downarrow} s'$ and s is positive or s' negative. We know that $s \leq_S s'$ hence there is a sequence $s \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow s'$ where all the s_i are internal in S. Assume for instance s is positive (the other case is similar). By courtesy of σ, we must have $n = 0$ and $\sigma s \rightarrow \sigma s'$.

Receptivity. Let $x \in \mathscr{C}(S_\downarrow)$ such that $\sigma_\downarrow x$ extends by a negative a. Since $\sigma([x]_S) = \sigma(x)$, by receptivity of σ, there exists an extension $s \in S$ of $[x]_S$ (hence s is an extension of x) such that $\sigma s = a$. Assume there is another extension of x, $s' \in S_\downarrow$ with $\sigma s = a$. As s' is also an extension of $[x]_S$, $s = s'$ by receptivity of σ. □

EXAMPLE 2.63. Covered strategies featuring conflict are not instances of uncovered strategies as their conflict occurs between visible events (contradicting the secrecy condition). Such visible events need to be prefixed by a neutral event that will be the source of the conflict. For instance, the strategy `sum` of Example 2.4 is represented by the following *uncovered* strategy:

$$a : \mathbf{proc}, \quad b : \mathbf{proc} \quad \vdash \quad \mathtt{sum}\, a\, b \quad : \mathbf{proc}$$



Neutral events of `sum` are drawn on a particular component even though mathematically, they do not belong to any.

Uncovered strategies are stable under composition:

LEMMA 2.64. *Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be uncovered strategies. Then $\tau \circledast \sigma$ is an uncovered strategy.*

PROOF. *Courtesy.* Assume $p \rightarrow_{T \circledast S} p'$ and for instance that p is positive (the case p' negative is similar). By Lemma 2.45, we have that either $\Pi_1 p \rightarrow_{S \| C} \Pi_1 p'$ or $\Pi_2 p \rightarrow_{A \| T} \Pi_2 p'$. If $(\tau \circledast \sigma)p \rightarrow_{A \| C} (\tau \circledast \sigma)p'$ (with both defined) does not hold, then $\Pi_1 p \rightarrow_S \Pi_1 p'$ or $\Pi_2 p \rightarrow_T \Pi_2 p'$. Both contradict the courtesy of σ and τ.

Receptivity. Let $w \in \mathscr{C}(T \circledast S)$ such that $(\tau \circledast \sigma)w$ can be extended by a negative c, with $c \in C$ for instance. Write $\Pi_2 z = x_A \parallel x_T$ with $x_A \in \mathscr{C}(A)$, and $x_T \in \mathscr{C}(T)$.

The hypothesis implies that τx_T can also extend by c, and by receptivity there exists a unique extension $t \in T$ of x_T with $\tau t = c$. The configuration w correspond to the interaction state $(\Pi_1 w, \Pi_2 w)$. Since $(\Pi_1 w \cup \{(2, \tau t)\}, \Pi_2 w \cup \{(2, t)\})$ is a valid interaction state, by Lemma 2.44 there exists an extension p of z with $(\tau \circledast \sigma)(p) = c$. Uniqueness follows directly from those of σ and τ.

Secrecy. Assume we have $z \in \mathscr{C}(T \circledast S)$ with incompatible extensions p_1 and p_2. By Lemma 2.10, this amounts to an incompatible extension either in S or in T: and then we can conclude by secrecy of σ and τ. □

4.3.1. *Interaction of uncovered strategies.* To conclude the section, we provide some terminology and results to work on the interaction of uncovered strategies. Courtesy and receptivity impose some structure on this operation which is otherwise very loose. Those results will be key when reasoning on interaction of strategies. Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be uncovered strategies. First, events of $\tau \circledast \sigma$ split in three disjoint categories:

- external-Opponent moves: those moves $s \in \tau \circledast \sigma$ that are mapped to a negative move of $A^\perp \parallel C$,
- σ-action: those moves $s \in \tau \circledast \sigma$ such that $\Pi_1 s$ is defined and a nonnegative element of S,
- τ-action: those moves $t \in \tau \circledast \sigma$ such that $\Pi_2 t$ is defined and a nonnegative element of T.

A **σ-move** is a move of $T \circledast S$ on which Π_1 is defined, and similarly a **τ-move** is a move of $T \circledast S$ on which Π_2 is defined. Using courtesy and receptivity, we can strengthen properties on the interaction.

LEMMA 2.65 (Interaction toolbox). *Let $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A^\perp$ be courteous and receptive maps of event structures.*

(1) For $p \in S \wedge T$, either $\Pi_1 p$ or $\Pi_2 p$ is nonnegative (but not both).
(2) For $p \rightarrowtriangle p' \in S \wedge T$ such that $\Pi_1 p'$ is nonnegative, then $\Pi_1 p \rightarrowtriangle \Pi_1 p'$
(3) For p, p' incompatible extensions of $x \in \mathscr{C}(S \wedge T)$ then either $\Pi_1 p$ and $\Pi_1 p'$ are nonnegative and incompatible extensions of $\Pi_1 x$ or $\Pi_2 p$ and $\Pi_2 p'$ are nonnegative and incompatible extensions.

Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be courteous and receptive maps of event structures. Their interaction $\tau \circledast \sigma$ satisfies the following properties:

(4) For $p \in T \circledast S$, at least $\Pi_1 p \in S$ or $\Pi_2 p \in T$. If both are satisfied, then p is a synchronized event on B.
(5) If $p \rightarrowtriangle p'$ and p' is a σ-action, then $\Pi_1 p \rightarrowtriangle_S \Pi_1 p'$
(6) If $x \in \mathscr{C}(T \circledast S)$ extends incompatibly by p_1 and p_2, then either $\Pi_1 x \cap S^\gamma$ extends incompatibly in S by $\Pi_1 p_1$ and $\Pi_2 p_2$, or $\Pi_2 x \cap T$ extends incompatibly in T by $\Pi_2 p_1$ and $\Pi_2 p_2$.

This lemma states that immediate causalities and conflict can always be traced back to S or T.

PROOF. *(1)* Clear: both projections cannot be negative.

(2) By Lemma 2.45, we have either $\Pi_1 p \rightarrowtriangle \Pi_1 p'$ or $\Pi_2 p \rightarrowtriangle \Pi_2 p'$. If the latter case is true, then given that $\Pi_2 p'$ will be negative in T, by courtesy it follows that $\tau(\Pi_2 p) \rightarrowtriangle \tau(\Pi_2 p')$. This implies that $\Pi_1 p \leq \Pi_1 p'$ since σ is a map of event

[γ] That is, the subset of $\Pi_1 x$ containing only events in the S component of $S \parallel C$

structures and thus reflects the causal order. If there was $\Pi_1 p < s < \Pi_1 p'$, this would contradict $p \twoheadrightarrow p'$ since Π_1 is also a map of event structures.

(3) Reasoning similar to (2). A minimal conflict originates in either S or T by Lemma 2.45. If it originates in T, then by receptivity, it must be originally present in the game, and hence S must have it as well.

(4-7) Those points are direct consequence of the previous points. □

4.3.2. *Invariance under copycat.* We now move on to show that $\mathfrak{a}_A \circledast \sigma$ is weakly bisimilar to σ. To show that, we need a good grasp on $\mathfrak{a}_A \circledast \sigma$. In particular, it is deadlock-free, meaning that all bijections are secured:

LEMMA 2.66. *Let $\tau : T \rightharpoonup A^\perp \parallel B$ be an uncovered pre-strategy satisfying: if $t \leq t'$ and both t and t' are sent by τ to $A^\perp$, then $\tau t \leq \tau t'$.*

Write $\tau_A : T \rightharpoonup A^\perp$ and $\tau_B : T \rightharpoonup B$ for the induced partial maps. For any uncovered pre-strategy $\sigma : S \rightharpoonup A$, and configurations x of S and y of T with $\sigma x = \tau_A y$, the induced bijection $x \parallel y_\star \parallel \tau_B y \simeq x_\star \parallel y$ is secured.

As a consequence, we have an order isomorphism:

$$\mathscr{C}(T \circledast S) \cong \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(T) \mid \sigma x = \tau_A y\}$$

PROOF. Assume that the bijection is not secured. Without loss of generality, there is a causal loop of the form $(v_1, t_1) \lhd \ldots \lhd (v_{2n}, t_{2n})$ such that $t_{2i} < t_{2i+1}$ and $v_{2i+1} < v_{2i+2}$ and $t_{2n} < t_1$. Since $t_{2i} < t_{2i+1}$, both (v_{2i}, t_{2i}) and (v_{2i+1}, t_{2i+1}) do not project to neutral elements of $S \parallel C$. Similarly, they do not project to neutral elements of T: they must be visible. In particular $v_i \in S \parallel B$.

Assume that $v_{2i+1} \in B$. Then $v_{2i+2} \in B$ and we have that $\tau(t_{2i+1}) = v_{2i+1} \leq v_{2i+2} = \tau(t_{2i+2})$. By Lemma 2.12, it follows that $t_{2i+1} \leq t_{2i+2}$. If the only two steps of the causal loop were (v_{2i+1}, t_{2i+1}) and (v_{2i}, t_{2i}), we have a loop in T and a contradiction. Otherwise, we can remove the steps $2i + 1$ and $2i + 2$ and keep a causal loop. Removing them, if there is a loop of length one remaining, then we have a direct contradiction (*i.e.* $t_1 < t_1$). Otherwise without loss of generality we can assume $v_i \in S$ for every i. In this case, by hypothesis on τ we have that $t_{2i} < t_{2i+1}$ implies that $\sigma v_{2i} = \tau t_{2i} < \tau t_{2i+1} = \sigma v_{2i+1}$. By Lemma 2.12 again, it follows that $v_1 < \ldots < v_1$ – a contradiction.

This establishes that the bijection induced by any pair of synchronized configurations (w, y) is secured and thus is a configuration of the interaction. □

Since copycat satisfies the axioms for τ of Lemma 2.66, we have the following isomorphism for any uncovered $\sigma : S \rightharpoonup A$, after simplification:

$$\mathscr{C}(\mathfrak{C}_A \circledast S) \cong \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(A) \mid \sigma x \parallel y \in \mathscr{C}(\mathfrak{C}_A)\}$$

Given such a pair (x, y) we write $\langle x, y \rangle$ for the corresponding configuration of $\mathfrak{C}_A \circledast S$. Remember that from Lemma 2.44, its events correspond to elements of the graph of corresponding secured bijection. Those events are of two kinds: internal events of the shape $(s, -)$ with $s \in x$ and external events of the shape $(-, a)$ with $a \in y$. This coercion will be kept implicit to ease the proofs. We can now prove the main result.

PROPOSITION 2.67. *For an uncovered strategy $\sigma : S \rightharpoonup A$, $\mathfrak{a}_A \circledast \sigma \approx \sigma$.*

PROOF. We define a bisimulation between σ and $\mathfrak{a}_A \circledast \sigma$ as follows:

$$\mathcal{R} = \{(x, \langle x', \sigma x \rangle) \in \mathscr{C}(S) \times \mathscr{C}(\mathfrak{C}_A \circledast S) \mid x' \sqsubseteq_S x\}.$$

Remember the Scott order $\sqsubseteq_S$ from Section 1.2.4. Notice that if $(x, \langle x', \sigma x\rangle) \in \mathcal{R}$, then $x \cup x' \in \mathscr{C}(S)$, as otherwise there would be a minimal conflict with a positive event, which is not possible for an uncovered strategy.

We now prove $\mathcal{R}$ is a weak bisimulation: assume $(x, \langle x', \sigma x\rangle) \in \mathcal{R}$.

Assume $x \overset{s}{\text{---}\!\subset}$. There are three cases:

- s is negative: then $\langle x', \sigma x\rangle \overset{(-,\sigma s)}{\text{---}\!\subset} \langle x', \sigma x \cup \{\sigma s\}\rangle$ and
$$(x \cup \{s\}, (x', \sigma x \cup \{\sigma s\})\}) \in \mathcal{R}$$
- s is positive: then $x' \cup [s] \in \mathscr{C}(S)$. Otherwise, s (or one of its predecessors) would be conflicting with a positive event in x' (since $x \cap x' \subseteq^+ x'$) contradicting the fact that σ is an uncovered strategy. Hence:
$$\langle x', \sigma x\rangle \subseteq^\star \langle x' \cup [s], \sigma x\rangle \overset{(-,\sigma s)}{\text{---}\!\subset} (x' \cup [s], \sigma x \cup \{\sigma s\}),$$
which implies $(x \cup \{s\}, \langle x' \cup [s], \sigma x \cup \{\sigma s\}\rangle) \in \mathcal{R}$.
- s is neutral: by a similar argument $x \cup x' \cup [s] \in \mathscr{C}(A)$ (otherwise neutral s would conflict with an event in $x \setminus x'$ which are all visible). It follows that $(x', y) \subseteq^\star \langle x' \cup [s], \sigma x\rangle$ and
$$(x \cup \{s\}, \langle x' \cup [s], \sigma x\rangle) \in \mathcal{R}$$

Assume now that $\langle x', \sigma x\rangle \overset{p}{\text{---}\!\subset}$. We proceed by case distinction on p:

- $(-, a^-)$: since $\sigma x \overset{a}{\text{---}\!\subset}$, by receptivity there exists $s \in S$ with $x \overset{s}{\text{---}\!\subset}$ and $\sigma s = a$. We have $(x \cup \{s\}, \langle x', y \cup \{a\}\rangle) \in \mathcal{R}$.
- $(-, a^+)$: Since $\sigma(x'_\downarrow) \parallel \sigma x \in \mathfrak{CC}_A$ and $\sigma x \overset{a}{\text{---}\!\subset}$, $a \in \sigma(x'_\downarrow)$ and there exists $s \in x'$ with $\sigma s = a$. Since $x \cup x' \in \mathscr{C}(A)$, it follows that $x \cup [s] \in \mathscr{C}(S)$. Moreover $x \cup [s] = x \cup \{s\}$, because by courtesy immediate dependencies of s are negative or neutral (or in the game) so must be in x. Finally
$$(x \cup [s], \langle x', \sigma x \cup \{a\}\rangle) \in \mathcal{R}.$$
- $(s, -)$: as above, we have that $x \cup [s] \in \mathscr{C}(S)$ and moreover by courtesy $x \subseteq^\star x \cup [s]$. Hence:
$$(x \cup [s], \langle x' \cup \{s\}, \sigma x\rangle \in \mathcal{R}) \quad \square$$

To conclude, we need a well-behaved notion of uncovered strategies from A to B, generalizing Lemma 2.32:

LEMMA 2.68. *Let $\sigma : A \nrightarrow B$ be an uncovered pre-strategy. There is an isomorphism*
$$\mathfrak{a}_B \circledast \sigma \circledast \mathfrak{a}_A \cong \mathfrak{a}_{A^\perp \parallel B} \circledast \sigma.$$
As a result, the following are equivalent:

(1) σ is an uncovered strategy on $A^\perp \parallel B$ (ie. $\mathfrak{a}_{A^\perp \parallel B} \odot \sigma \approx \sigma$),
(2) σ satisfies $\mathfrak{a}_B \circledast \sigma \circledast \mathfrak{a}_A \approx \sigma$.

PROOF. We prove this by building an isomorphism: $\mathfrak{a}_B \circledast \sigma \circledast \mathfrak{a}_A \cong \mathfrak{a}_{A^\perp \parallel B} \circledast \sigma$. First, remember that we have
$$\mathfrak{a}_B \star \sigma \star \mathfrak{a}_A \cong (\mathfrak{a}_A \parallel B \parallel B) \wedge (A \wedge \sigma \wedge B) \wedge (A \parallel A \parallel \mathfrak{a}_B)$$
$$\mathfrak{a}_{A^\perp \parallel B} \cong \mathfrak{CC}_{A^\perp \parallel B} \star (\sigma \parallel A \parallel B)$$

By proving that $\mathbb{C}_A \parallel \mathbb{C}_B$ satisfies the corresponding universal property, it is direct to show that

$$\mathbb{C}_A \parallel \mathbb{C}_B \cong (\mathbb{C}_A \parallel B \parallel B) \wedge (A \parallel A \parallel \mathbb{C}_B)$$

As a result, we have

$$\alpha_B \star \sigma \star \alpha_A \cong (\mathbb{C}_B \parallel \mathbb{C}_A) \wedge (\sigma \parallel A \parallel B).$$

Moreover, we have the following isomorphism:

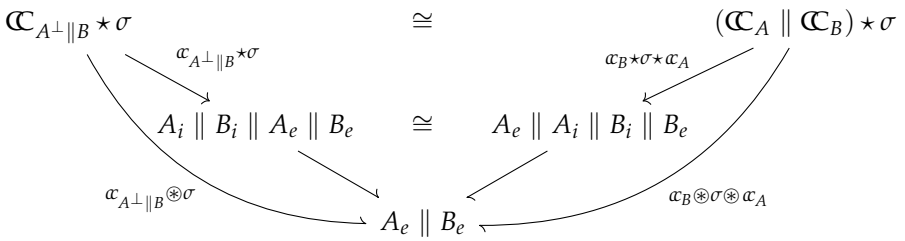
$$\begin{array}{ccc} \sigma \parallel A \parallel B & \cong & A \parallel \sigma \parallel B \\ \downarrow & & \downarrow \\ A_i \parallel B_i \parallel A_e \parallel B_e & \cong & A_e \parallel A_i \parallel B_i \parallel B_e \end{array}$$

The subscript at the bottom (i for internal and e for external) are just here to spell out the isomorphism. Similarly, for copycat:

$$\begin{array}{ccccc} \mathbb{C}_{A^\perp \parallel B} & \cong & \mathbb{C}_{A^\perp} \parallel \mathbb{C}_B & \cong & \mathbb{C}_A \parallel \mathbb{C}_B \\ \downarrow & & \downarrow & & \downarrow \\ A_i \parallel B_i \parallel A_e \parallel B_e & \cong & A_i \parallel A_e \parallel B_i \parallel B_e & \cong & A_e \parallel A_i \parallel B_i \parallel B_e \end{array}$$

Notice that moving from $\mathbb{C}_{A^\perp}$ to $\mathbb{C}_A$ consists in permuting its codomain.

By combining the two previous diagram, we get the following diagram:



from which the conclusion follows. □

At last, we have everything needed to build a category:

PROPOSITION 2.69. *The following is a category* $\mathrm{CG}^{\approx}_{\circledast}$*:*

Objects: *Race-free games,*
Morphisms from A **to** B**:** *uncovered strategies* $A \dashv\!\!\succ B$ *up to* $\approx$*,*
Composition: *composition of uncovered strategies,*
Copycat: *the usual copycat.*

This category is also compact-closed, but we omit the proofs: this category is built as an intermediary step on the way to essential strategies but will not be used in the rest of the development.

5. Essential strategies

The category built in the previous section can be used to give interpretations of concurrent and nondeterministic languages, sound for must-equivalence and other testing equivalences. However no hiding is performed so the interpretation of terms grows rapidly with the size of the term, even for closed terms of base types. Moreover, weak bisimulation is hard to decide in practice. In this section, we propose a setting that has the best of both worlds:

- we do not hide divergences during the composition to remain sound with respect to must-equivalence,
- we hide "synchronization events" that occur during the composition with copycat to get a category up to isomorphism.

5.1. Essential events. To do that, we need to tell apart harmless synchronization events from divergences. This is done through the notion of **essential events**:

DEFINITION 2.70 (Essential events). Let $\sigma : S \rightharpoonup A$ be an uncovered pre-strategy. An internal event $s \in S$ is **essential** when it is involved in a minimal conflict, that is there exists $x \in \mathscr{C}(S)$ that can extend by s and another event $s' \in S$ such that $x \cup \{s, s'\} \notin \mathscr{C}(S)$.

Write E_S for the set of all visible or essential events of an event structure with partial polarity S, and $\mathscr{E}(S)$ for $S \downarrow E_S$ for the essential part of S. Finally, if $\sigma : S \rightharpoonup A$ is a pre-strategy, write $\mathscr{E}(\sigma) : \mathscr{E}(S) \rightharpoonup A$ for its essential part. Hiding non-essential events does not change the behaviour up to weak bisimulation:

LEMMA 2.71. *For $\sigma : S \rightharpoonup A$ an uncovered strategy, we have $\sigma \approx \mathscr{E}(\sigma)$.*

PROOF. Define $\mathcal{R} = \{(x, x \cap E_S) \mid x \in \mathscr{C}(S)\}$. We prove it is a bisimulation. Let $(x, x \cap E_S) \in \mathcal{R}$.

Extensions of x. Assume x can extend by an event s. Either $s \in E_S$ and then $x \cap E_S$ also extends by s in $S \downarrow E_S$ and $(x \cup \{s\}, x \cap E_S \cup \{s\}) \in \mathcal{R}$ as desired, or $s \notin E_S$ (then s is internal) and $(x \cup \{s\}, x \cap E_S) \in \mathcal{R}$.

Extensions of $x \cap E_S$ Assume $x \cap E_S$ can be extended by an event $s \in E_S$, and $x \cup [s] \notin \mathscr{C}(S)$. Since it is downclosed, it must be that $x \cup [s] \notin \mathrm{Con}_S$. Since x and $[s]$ are consistent, there must exist $x' \subseteq x \cup [s]$ with two incompatible extensions $s_0 \in x$ and $s_1 \in [s]$. By definition, s_0 and s_1 are essential. This is absurd because this would mean that $(x' \cap E_S) \cup \{s_0, s\} \subseteq x \cap E_S \cup \{s\} \in \mathscr{C}(S \downarrow E_S)$ but this set is not consistent in S (hence not consistent in $S \downarrow E_S$). Having just proved that $x \cup [s] \in \mathscr{C}(S)$, the conclusion follows as $(x \cup [s], \{s\} \cup (x \cap E_S)) \in \mathcal{R}$. □

The operation $\mathscr{E}(\cdot)$ preserves uncovered strategies:

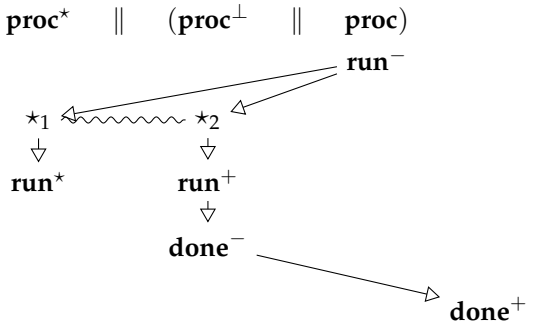
LEMMA 2.72. *For $\sigma : S \rightharpoonup A$ an uncovered strategy, $\mathscr{E}(\sigma)$ is an uncovered strategy.*

PROOF. Straightforward. □

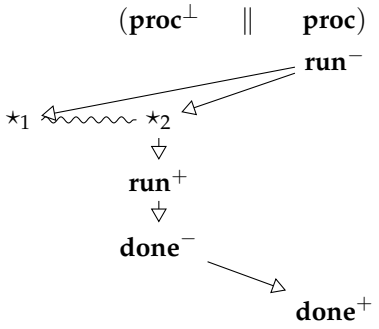
5.1.1. *Another composition of uncovered strategies.* Given uncovered pre-strategies $\sigma : A \mathrel{-\!\!\!\vdash\!\!\!\succ} B$ and $\tau : B \mathrel{-\!\!\!\vdash\!\!\!\succ} C$, define a new composition that hides inessential events:

$$\tau \circledcirc \sigma = \mathscr{E}(\tau \circledast \sigma).$$

EXAMPLE 2.73. In Example 2.55, we showed that $\mathtt{sum} \odot \bot$ is isomorphic to copycat, as the divergence is hidden. In this setting, the interaction $\mathtt{sum} \circledast \bot$ becomes (the uncovered strategy for sum is given in Example 2.63):



The only minimal conflict is between $\star_1$ and $\star_2$, hence $\mathtt{sum} \circledcirc \bot$ is:



where the only inessential event $\mathbf{run}^\star$ was hidden.

Isomorphism and this new composition $\circledcirc$ interact well together:

LEMMA 2.74. *Isomorphism is a congruence for $\circledcirc$ and $\circledcirc$ is associative up to $\cong$.*

PROOF. Let $\sigma : S \rightharpoonup A^\perp \parallel B$, $\sigma' : S' \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$, $\tau' : T' \rightharpoonup B^\perp \parallel C$ such that there exists $\varphi : \sigma \cong \sigma'$ and $\psi : \tau \cong \tau'$. Using the universal property of the interaction we build an isomorphism:

$$\langle (\varphi \parallel C) \circ \Pi_1, (A \parallel \psi) \circ \Pi_2 \rangle : (S \parallel C) \wedge (A \parallel T) \cong (S' \parallel C) \wedge (A \parallel T')$$

This isomorphism preserves essential events hence restricts to an isomorphism

$$\tau \circledcirc \sigma \cong \tau' \circledcirc \sigma'.$$

For associativity, the proof is delayed to Section 6. □

To get a category up to isomorphism, as before, we need to study the composition of uncovered strategies with copycat.

5.2. Composition with copycat. In this section, we fix an uncovered strategy $\sigma : S \rightharpoonup A$ and study the composition $\mathfrak{a}_A \circledcirc \sigma$. Remember that through Lemma 2.66, configurations of the interaction $\mathfrak{a}_A \circledast \sigma$ have a simple description:

$$\mathscr{C}(\mathfrak{C}_A \circledast S) \cong \{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(A) \mid \sigma x \parallel y \in \mathscr{C}(\mathfrak{C}_A)\}.$$

As before, we write $\langle x, y\rangle \in \mathscr{C}(\mathbb{C}_A \circledast S)$ for suitable $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(\mathbb{C}_A)$. By Lemma 2.24, configurations of $\mathbb{C}_A \circledcirc S$ correspond to those of $\mathbb{C}_A \circledast S$ whose maximal events are visible or essential: this correspondence is left implicit.

Our result relies on an understanding of essential events of $\mathbb{C}_A \circledcirc S$. First, the following lemma clarifies minimal conflict in $\mathbb{C}_A \circledcirc S$ as originating from essential events in S or already present in the game:

LEMMA 2.75. *Let $z \in \mathscr{C}(\mathbb{C}_A \circledast S)$ with two incompatible extensions p, p'. At least one of the following two conditions are true:*

(1) $\Pi_1 p, \Pi_1 p' \in S$ are both essential in S,

(2) $\Pi_1 p, \Pi_1 p' \in A$ are both negative with $(\mathbb{a}_A \circledast \sigma)(z \cup \{p, p'\}) \notin \mathscr{C}(A^\perp \parallel A)$.

PROOF. We apply Lemma 2.45: at least one of $\Pi_1(z \cup \{p, p'\})$ and $\Pi_2(z \cup \{p, p'\})$ is not consistent. Since the second case is equivalent to $(\mathbb{a}_A \circledast \sigma)(z \cup \{p, p'\}) \notin \mathscr{C}(A^\perp \parallel A)$, we have the following two cases:

- $(\mathbb{a}_A \circledast \sigma)(z \cup \{p, p'\}) \in \mathscr{C}(A^\perp \parallel A)$: it follows that $\Pi_1(z \cup \{p, p'\}) \notin \mathscr{C}(S)$ hence $\Pi_1 p, \Pi_1 p' \in S$. Since the projection on the game is consistent, $\Pi_1 p$ and $\Pi_1 p'$ must be neutral and hence essential: we proved (1).
- If $\mathbb{a}_A \circledast \sigma(z \cup \{p, p'\})$ is not consistent in $A^\perp \parallel A$. Hence $\Pi_2 z$ has two incompatible extensions $\Pi_2 p, \Pi_2 p'$ in $\mathbb{C}_A$ and $\Pi_2 p$ and $\Pi_2 p$ are negative in $A^\perp \parallel A$. This means that $\Pi_1 x$ has two incompatible *positive* extensions $\Pi_1 p$ and $\Pi_1 p'$. Since σ is an uncovered strategy, $\Pi_1 p$ and $\Pi_1 p'$ cannot live in S: they have to live in A: we proved (2). □

From that follows the key lemma:

LEMMA 2.76. *For an essential $s \in S$, $\langle [s], \sigma[s]\rangle$ is a prime configuration $[\mathfrak{p}s]$ of $\mathbb{C}_A \circledcirc S$. The mapping $s \in S \mapsto \mathfrak{p}s$ is a one-to-one correspondence between essential events of S and those of $\mathbb{C}_A \circledcirc S$.*

PROOF. *$\langle [s], \sigma[s]\rangle$ is prime.* Let $s \in S$ be an essential event. Remember that events of the configuration $\langle [s], \sigma[s]\rangle$ correspond to pairs of the form $(s_0, -)$ or $(-, \sigma s_0)$. Assume $(-, \sigma s_0)$ is maximal in $\langle [s], \sigma[s]\rangle$. If s_0 is negative, then we would have $(-, \sigma s_0) < (s_0, \sigma s_0)$ (causal link induced by copycat). If s_0 is positive, then there exists $s_0 \twoheadrightarrow s_1 \leq s$. By courtesy, s_1 is visible, and we have $(s_0, \sigma s_0) < (s_1, \sigma s_1)$ (induced again by copycat). As a result maximal events of $\langle [s], \sigma[s]\rangle$ are of the form $(s_0, -)$ with $s_0 \leq s$. It is easy to see that for $s_0 \leq s$, we have $(s_0, -) \leq (s, -)$. This implies that $(s, -)$ must be the top element of $\langle [s], \sigma[s]\rangle$. This means that $\langle [s], \sigma[s]\rangle$ is indeed a prime configuration $[\mathfrak{p}s]$.

$\mathfrak{p}s$ is essential. Let $s \in S$ be an essential event, and let $x \in \mathscr{C}(S)$ and $s' \in S$ such that s and s' are incompatible extensions of x. By definition of $\mathbb{C}_A \circledcirc S$, it is enough to show that $\mathfrak{p}s$ is essential in $\mathbb{C}_A \circledast S$. The configuration $\langle x, \sigma x\rangle \in \mathscr{C}(\mathbb{C}_A \circledast S)$ extends by both $\mathfrak{p}s$ and $\mathfrak{p}s'$. If these extensions were compatible, $\Pi_1(\langle x, \sigma x\rangle \cup \{\mathfrak{p}s, \mathfrak{p}s'\}) = x \cup \{s, s'\}$ would be a configuration of S: absurd. Hence, $\mathfrak{p}s$ must be essential. As a consequence, the mapping $s \mapsto \mathfrak{p}s$ is indeed well-defined. Its injectivity is straightforward.

Surjectivity of $s \mapsto \mathfrak{p}s$. Let p be an essential event of $\mathbb{C}_A \circledcirc S$. Since p is also essential in $\mathbb{C}_A \circledast S$, there exist $w \in \mathscr{C}(\mathbb{C}_A \circledast S)$ and $p' \in \mathbb{C}_A \circledast S$ such that p and p' are incompatible extensions of w. First, notice that Π_1 is actually total since $\mathbb{C}_A$ does not contain any internal events: hence $\Pi_1 w$ extends incompatibly by $\Pi_1 p$ and $\Pi_1 p'$. This means that $\Pi_1 p$ is essential, and $p = \mathfrak{p}(\Pi_1 p)$. □

From this result, we fully characterize $\mathfrak{a}_A \circledcirc \sigma$:

LEMMA 2.77. *Let $\sigma : S \rightharpoonup A$ be an uncovered strategy. Then $\mathfrak{a}_A \circledcirc \sigma \cong \mathscr{E}(\sigma)$*

PROOF. Since σ is an uncovered strategy, $\sigma_\downarrow$ is a covered strategy and there exists $\varphi : S_\downarrow \cong \mathbb{C}_A \odot S_\downarrow$ by Theorem 2.38. We now show how to extend this to

$$\bar{\varphi} : S \downarrow E_S \cong \mathbb{C}_A \circledcirc S.$$

By Lemma 2.76, for essential $s \in S$, we define $\bar{\varphi}(s) = \mathfrak{p}s \in \mathbb{C}_A \circledcirc S$. The same lemma implies that $\bar{\varphi}$ is a bijection, indeed $\mathbb{C}_A \circledcirc S$ contains only visible events (in the range of φ) and essential events (of the form $\langle [s], \sigma[s] \rangle$ with essential $s \in S$). Both $\bar{\varphi}$ and its inverse are easily shown to be maps of event structures. □

This lemma calls for a natural definition of **essential strategy**:

DEFINITION 2.78 (Essential strategy). An **essential strategy** σ is an uncovered strategy where all internal events are essential (equivalently, such that $\mathscr{E}(\sigma) \cong \sigma$).

Lemma 2.77 can be turned into an equivalence, generalizing the result of [**RW11**].

THEOREM 2.79. *Let $\sigma : S \rightharpoonup A$ be an uncovered pre-strategy. It is an essential strategy if and only if $\mathfrak{a}_A \circledcirc \sigma \cong \sigma$.*

PROOF. By Lemma 2.77, the direct implication is clear. Assume there exists $\varphi : S \cong \mathbb{C}_A \circledcirc S$. Since $\mathscr{E}(\cdot)$ is idempotent, $\mathscr{E}(\sigma) \cong \sigma$. By Lemma 2.72, it is enough to show that $\mathbb{C}_A \circledast \sigma$ is an uncovered strategy.

Courtesy: Assume $p^+ \rightarrowtriangle p'$ in the interaction $\mathbb{C}_A \circledast S$. By Lemma 2.45, and since p is positive, we have $\Pi_2 p \rightarrowtriangle_{\mathbb{C}_A} \Pi_2 p'$ and we conclude by courtesy of $\mathfrak{a}_A$. A similar reasoning applies for immediate causal links $p \rightarrowtriangle p'^-$ in $\mathbb{C}_A \circledast S$.

Receptivity: Assume a configuration (x, y) in $\mathbb{C}_A \circledcirc S$ such that y extends by a negative $a \in A$. Then clearly $(x, y \cup \{a\}) \in \mathbb{C}_A \circledcirc S$. Uniqueness is easy to check.

Secrecy: Assume $\langle x, y \rangle \in \mathscr{C}(\mathbb{C}_A \circledast S)$ with two incompatible extensions p_1 and p_2. By Lemma 2.65, either x extends incompatibly in S by $\Pi_1 p_1$ and $\Pi_2 p_2$ or $\sigma x \parallel y$ extends incompatibly in $\mathbb{C}_A$ by $\Pi_2 p_1$ and $\Pi_2 p_2$. In the first, case, this conflict is a conflict between internal events, and in the second case a conflict between negative events (by secrecy of copycat). □

Since isomorphisms preserve essential events, we deduce from Lemma 2.68 that $\mathfrak{a}_B \circledcirc \sigma \cong \sigma \circledcirc \mathfrak{a}_A$ for an essential strategy $\sigma : S \rightharpoonup A^\perp \parallel B$. Putting it altogether, we get a category of essential strategies:

PROPOSITION 2.80. *There is a compact-closed category $\mathrm{CG}^{\cong}_{\circledcirc}$ of:*

Objects: *race-free games,*
Strategies from A to B: *Essential strategies $A \multimap\!\!\!\!\blacktriangleright B$ up to isomorphism,*
Composition: *the composition operator $\circledcirc$,*
Identity: *copycat.*

PROOF. See Section 6. □

Hiding defines a functor $_\downarrow : \mathrm{CG}^{\cong}_{\circledcirc} \to \mathrm{CG}^{\cong}_{\odot}$, however the natural converse operation mapping a covered strategy σ to the essential strategy $\mathfrak{a}_A \circledcirc \sigma$ does not define a functor from $\mathrm{CG}^{\cong}_{\odot}$ to $\mathrm{CG}^{\cong}_{\circledcirc}$ but simply a lax-functor. (As the image of $\mathtt{sum} \odot \bot$ is only included in $(\mathtt{sum} \circledcirc \mathfrak{a}) \circledcirc (\bot \circledcirc \mathfrak{a})$).

This functor is not faithful in general (since it hides essential events), but on *deterministic strategies*, it is:

DEFINITION 2.81. An uncovered strategy $\sigma : S \rightharpoonup A$ is **deterministic** when all incompatible extensions s_1, s_2 of a configuration $x \in \mathscr{C}(S)$ are negative.

LEMMA 2.82. *An essential strategy $\sigma : S \rightharpoonup A$ is deterministic if and only if $\downarrow \sigma \cong \sigma$. As a result, $\downarrow$: $\mathrm{CG}^{\cong}_{\circledcirc} \to \mathrm{CG}^{\cong}_{\odot}$ is faithful on deterministic strategies.*

PROOF. Straightforward. □

Since our games are race-free, ϖ_A is deterministic [**Win12**].

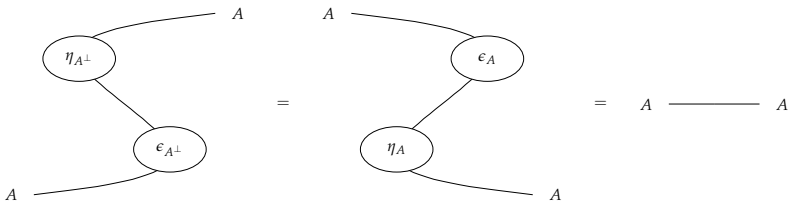
In the next chapter, we build on $\mathrm{CG}^{\cong}_{\circledcirc}$ a new compact-closed category, $\sim$-$\mathrm{tCG}^{\cong\!\cong}_{\circledcirc}$ which allows for non-linearity through symmetry. We end this chapter on the proof of the categorical structure. (Proposition 2.69, Proposition 2.80)

6. Proof of categorical structure

In this section, we provide proofs of the following:

(1) that $\circledcirc$ is an associative operation ($\circledast$ was already proven associative),
(2) that $\|$ extends to a symmetric monoidal product $\otimes$ on uncovered and essential strategies,
(3) that the resulting monoidal structure is compact-closed with the duality operation on games $A \mapsto A^{\perp}$.

We recall that point (3) amounts to giving essential strategies $\eta_A : 1 \rightarrowtail\!\!\!\!\triangleright A^{\perp} \otimes A$ and $\epsilon_A : A^{\perp} \otimes A \rightarrowtail\!\!\!\!\triangleright 1$, that is essential strategies on $A^{\perp} \parallel A$, satisfying the following equations:

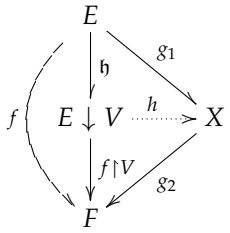


Picking copycat for both η_A and ϵ_A,we are left to prove a diagram between deterministic essential strategies which can be deduced from the compact-closure of $\mathrm{CG}^{\cong}_{\odot}$ via faithfulness of $\downarrow$ (See [**CCRW**] for detailed the proof for $\mathrm{CG}^{\cong}_{\odot}$). In this section, we only focus on the first two points.

6.1. Associativity of $\circledcirc$. We follow the same approach as in [**CCRW**]. The study of the ternary composition relies on partial maps. For an event structure E and $V \subseteq E$ there is a canonical partial map $\mathfrak{h} : E \to E \downarrow V$ defined as the (partial) identity on V. In fact, any partial map can be factored through such a map:

LEMMA 2.83. *Let $f : E \rightharpoonup F$ be a partial map, and V be the subset of events of E on which f is defined. Then, f factors as $(f \upharpoonright V) \circ \mathfrak{h}$ (where $f \upharpoonright V : E \downarrow V \to F$ is total). Moreover, for any other factorization $f = g_2 \circ g_1$ with $g_1 : E \rightharpoonup X$ and $g_2 : X \to F$, there is a unique total $h : E \downarrow V \to X$ such that $h \circ \mathfrak{h} = g_1$ and $g_2 \circ h = f \upharpoonright V$, as pictured in*

the diagram below:



We say that $\mathfrak{h} : E \rightharpoonup E \downarrow V$ *has the* ***partial-total universal property****.*

In **[CCRW]**, *hiding maps* are introduced as special case of partial maps:

DEFINITION 2.84 (Hiding maps). A partial map $f : E \rightharpoonup F$ is a **hiding map** when it satisfies one of the three equivalent conditions:

(1) Writing V for the domain of f, there is an isomorphism $\varphi : E \downarrow V \cong F$ with $\varphi \circ \mathfrak{h} = f$,
(2) The partial map f has the partial-total universal property,
(3) There exists a **hiding witness** for f that is a map $\mathrm{wit}_f : \mathscr{C}(F) \to \mathscr{C}(E)$ with $f \circ \mathrm{wit}_f(x) = x$ for $x \in \mathscr{C}(F)$ and $\mathrm{wit}_f \circ f(x) \subseteq x$ for $x \in \mathscr{C}(E)$.

This notion is very useful to establish that compositions of strategies are isomorphic as it reduces the problem to finding a hiding map between the interactions. Hiding maps are stable under composition, and well-behaved with respect to interaction:

LEMMA 2.85 (Zipping lemma). *Let* $\sigma : S \rightharpoonup A^\perp \parallel B, \sigma' : S' \rightharpoonup A^\perp \parallel B$ *and* $\tau : T \rightharpoonup B^\perp \parallel C$ *be uncovered pre-strategies. If* $\mathfrak{h} : S \rightharpoonup S'$ *is such that* $\tau \circ \mathfrak{h} = \sigma$*, there exists a hiding map* $T \circledast \mathfrak{h} : T \circledast S \rightharpoonup T \circledast S'$ *making the analogous triangle commute.*

PROOF. *Closed interaction.* Assume that $\sigma : S \rightharpoonup A, \sigma' : S' \rightharpoonup A, \tau : T \rightharpoonup A$ be uncovered pre-strategies and $\mathfrak{h} : S \rightharpoonup S'$ a hiding map with $\sigma = \sigma' \circ \mathfrak{h}$. Define

$$\mathfrak{h} \wedge T = \langle \mathfrak{h} \circ \Pi_1, \Pi_2 \rangle : S \wedge T \rightharpoonup S' \wedge T$$

via Lemma 2.47. From the witness function of $\mathfrak{h}$, it is straightforward to build a witness function for $\mathfrak{h} \wedge T$.

Open interaction. To prove the statement of the lemma, define $T \circledast \mathfrak{h}$ to be $(A \parallel T) \wedge (\mathfrak{h} \parallel C)$: this is the desired hiding map. □

We can conclude:

LEMMA 2.86. *Composition* $\circledcirc$ *of uncovered pre-strategies is associative.*

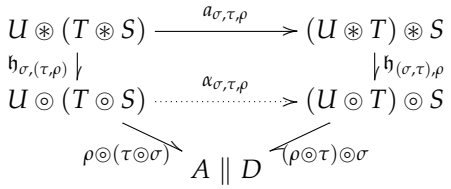
PROOF. Let $\sigma : S \rightharpoonup A^\perp \parallel B$, $\tau : T \rightharpoonup B^\perp \parallel C$ and $\rho : U \to C^\perp \parallel D$ be uncovered pre-strategies. We already know that the interaction is associative (Lemma 2.52), there is an isomorphism:

$$a_{\sigma,\tau,\rho} : U \circledast (T \circledast S) \cong (U \circledast T) \circledast S.$$

Applying Lemma 2.85 to the hiding maps $\mathfrak{h}_{\sigma,\tau} : T \circledast S \rightharpoonup T \circledcirc S$ and $\mathfrak{h}_{\tau,\upsilon} : U \circledast T \rightharpoonup U \circledcirc T$ gives the following hiding maps:

$$\mathfrak{h}_{\sigma,(\tau,\rho)} = (U \circledast T) \circledast S \xrightarrow{\mathfrak{h}_{\tau,\rho} \circledast S} (U \circledcirc T) \circledast S \xrightarrow{\mathfrak{h}_{\sigma,\rho \circledcirc \tau}} (U \circledcirc T) \circledcirc S$$
$$\mathfrak{h}_{(\sigma,\tau),\rho} = U \circledast (T \circledast S) \xrightarrow{U \circledast \mathfrak{h}_{\sigma,\tau}} U \circledast (T \circledcirc S) \xrightarrow{\mathfrak{h}_{\tau \circledcirc \sigma,\rho}} U \circledcirc (T \circledcirc S)$$

Associativity amounts to the commutation of the outer pentagon of this diagram:



The only thing to show is that the two maps have the same domain which follows from the fact that as $a_{\sigma,\tau,\rho}$ is an isomorphism, it preserves essential events (and visible events). By the partial-total factorization (Lemma 2.83), this diagram induces an isomorphism $\alpha_{\sigma,\tau,\rho} : (U \circledcirc T) \circledcirc S \to U \circledcirc (T \circledcirc S)$ as desired. □

This technology makes proving Lemma 2.53 elementary:

LEMMA 2.53. *For $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ uncovered pre-strategies,*

$$(\tau \circledast \sigma)_\downarrow \cong \tau_\downarrow \odot \sigma_\downarrow$$

PROOF. We want to establish an isomorphism:

$$((S \parallel C) \wedge (A \parallel T)) \downarrow V_{T \circledast S} \cong \big((S_\downarrow \parallel C) \wedge (A \parallel T_\downarrow)\big) \downarrow V_{T_\downarrow \odot S_\downarrow}$$

By Lemma 2.83, this amounts to finding a hiding map with domain $V_{T \circledast S}$

$$(S \parallel C) \wedge (A \parallel T) \rightharpoonup (S_\downarrow \parallel C) \wedge (A \parallel T_\downarrow) \downarrow V_{T_\downarrow \odot S_\downarrow}$$

We construct such a hiding map using Lemma 2.85 by composition:

$$T \circledast S \xrightarrow{\mathfrak{h}_T \circledast S} T_\downarrow \circledast S \xrightarrow{T_\downarrow \circledast \mathfrak{h}_S} T_\downarrow \circledast S_\downarrow \xrightarrow{\mathfrak{h}_{T_\downarrow} \circledast S_\downarrow} T_\downarrow \odot S_\downarrow$$

□

6.2. A symmetric monoidal product. To show that $\mathrm{CG}^{\cong}_{\circledcirc}$ is symmetric monoidal, we extend $\parallel$ from games to strategies as follows. Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup C^\perp \parallel D$ be uncovered pre-strategies. Define $\sigma \otimes \tau$ as

$$S \parallel T \xrightarrow{\sigma \parallel \tau} (A^\perp \parallel B) \parallel (C^\perp \parallel D) \cong (A \parallel C)^\perp \parallel (B \parallel D)$$

We use the notation $\sigma \otimes \tau$ to denote tensor product of strategies to distinguish it from $\sigma \parallel \tau$ which is just parallel composition at the level of maps: $\sigma \otimes \tau$ also involves reordering on games. It is easy to see that if σ and τ are essential strategies then so is $\sigma \otimes \tau$. This tensor operation also preserves determinism.

LEMMA 2.87. *The operation $\otimes$ is a functor $\mathrm{CG}^{\cong}_{\circledcirc} \to \mathrm{CG}^{\cong}_{\circledcirc}$.*

PROOF. Preservation of copycat follows from the same result holding in $\mathrm{CG}^{\cong}_{\odot}$ (see **[CCRW]**) and $\downarrow$ being faithful on deterministic strategies. Functoriality is proved in a similar way as associativity: prove the interactions are isomorphic and then use hiding maps to deduce the result on the level of composition. The key argument here once again is that essential events are stable under isomorphism. □

The unit for this operation will be the empty game denoted 1. To prove that this functor (along with its unit) is actually a symmetric monoidal operation, we need to define the corresponding structural morphisms. Such structural morphisms will be lifted from the category of event structure as $\parallel$ is already a symmetric monoidal operation there. Presentation is also taken from **[CCRW]**.

DEFINITION 2.88. Let A, B be games and let $f : A \to B$ a receptive, courteous map of event structures preserving polarities. Then, the map:

$$\begin{array}{rccl} \overline{f} & : & \mathfrak{C}_A & \to & A^\perp \parallel B \\ & & a & \mapsto & (A^\perp \parallel f) \circ \varpi_A(a) \end{array}$$

is a strategy called the **lifting** of f. Likewise, if $f : B^\perp \to A^\perp$ is receptive and courteous, we define its **co-lifting**:

$$\begin{array}{rccl} \overline{f}^\perp & : & \mathfrak{C}_B & \to & A^\perp \parallel B \\ & & c & \mapsto & (f \parallel B) \circ \varpi_B(c) \end{array}$$

Both liftings and co-liftings are deterministic uncovered strategies. The following key lemmas relate composition of strategies to lifted maps and composition of the corresponding maps in $\mathcal{E}$.

LEMMA 2.89. *Let $f : B \to C$ be a receptive courteous map of esps, and $\sigma : S \rightharpoonup A^\perp \parallel B$ be an essential strategy. Then, there is an isomorphism of strategies*

$$\overline{f} \circledcirc \sigma \cong (A^\perp \parallel f) \circ \sigma$$

Likewise, for $f : B^\perp \to A^\perp$ receptive courteous and $\sigma : S \to B^\perp \parallel C$ an essential strategy, there is an isomorphism of strategies:

$$\sigma \circledcirc \overline{f} \cong (f \parallel C) \circ \sigma$$

PROOF. We prove only the first case (the second one being similar). We see through Lemma 2.66 that $\varpi_B \circledast \sigma$ and $\overline{f} \circledast \sigma$ have isomorphic interactions, that we write $\mathfrak{C}_B \circledast S$. Its configurations can be described as

$$\{(x, y) \in \mathscr{C}(S) \times \mathscr{C}(B) \mid \sigma_2 x \parallel y \in \mathscr{C}(\mathfrak{C}_B)\}$$

where $\sigma_2 : S \rightharpoonup B$ is the obvious partial map, and the following diagram commutes:

$$\begin{array}{ccc} & \mathfrak{C}_B \circledast S & \\ {\scriptstyle \varpi_B \circledast \sigma}\swarrow & & \searrow{\scriptstyle \overline{f} \circledast \sigma} \\ A \parallel B \parallel B & \xrightarrow{A \parallel B \parallel f} & A \parallel B \parallel C \end{array}$$

as both arrows map $\langle x, y \rangle$ to $\sigma x \parallel fy$. Since essential and visible events are preserved, we get the following diagram:

$$\begin{array}{ccccc} S & & \xrightarrow{\cong} & & \mathfrak{C}_B \circledcirc S \\ {\scriptstyle \sigma}\downarrow & & & \swarrow{\scriptstyle \varpi_B \circledcirc \sigma} & \searrow{\scriptstyle \overline{f} \circledcirc \sigma} \\ A \parallel B & & \xrightarrow[A \parallel f]{} & & A \parallel C \end{array}$$

using the fact that σ is an essential strategy, hence $\sigma \cong \varpi_B \circledcirc \sigma$. We conclude by noticing the composite isomorphism $(A \parallel f) \circ \sigma \cong \overline{f} \circledcirc \sigma$. □

From this lemma, we deduce the compact-closure:

THEOREM 2.90. $\mathrm{CG}^{\cong}_{\circledcirc}$ *is compact-closed.*

PROOF. We know composition is associative, and that $\otimes$ is a bifunctor. The symmetric monoidal structure is lifted from $\mathcal{E}$: diagrams and naturality all follow from the corresponding diagrams and naturality in $\mathcal{E}$ and Lemma 2.89.

The dual of a game A is simply defined as $A^{\perp}$. We have two strategies:

$$\begin{array}{rcccl} \eta_A & : & \mathbb{C}_A & \to & 1^{\perp} \parallel (A^{\perp} \parallel A) \\ \epsilon_A & : & \mathbb{C}_A & \to & (A \parallel A^{\perp})^{\perp} \parallel 1 \end{array}$$

defined in the obvious way and satisfy the laws of a compact-closed category (as proved in **[CCRW]**). □

To prove that $\mathrm{CG}^{\cong}_{\circledast}$ is also compact-closed, one can slightly change the statement of Lemma 2.89 and conclude similarly – however we will not be needing this in the next chapters.

CHAPTER 3

Thin concurrent games

In the previous chapter, we introduced a compositional framework to describe strategies as event structures allowing for nondeterministic and concurrent behaviours. However, one major limitation of this framework is the linearity condition imposed on strategies (technically arising as the *local injectivity* condition on maps of event structures). Strategies can play moves at most once in an execution. Since function calls are represented by moves, in $\mathrm{CG}^{\cong}_{\circledcirc}$ only affine languages (languages where variables are used at most once) can be modelled.

This *local injectivity* condition is key to define the pullback in the category of event structures and composition of strategies. Without this condition, it is unknown how to define the pullback of maps, at the heart of the interaction of strategies. To circumvent this problem, we follow in this chapter another approach inspired from linear logic: a non-linear strategy on a game A will be seen as a linear strategy on a derived game $!A$ where moves of A have been duplicated: a single move from A yields infinitely many *copies* of it, explicitly labelled by a natural number, their *copy index*.

This duplication raises two difficult problems well-known in game semantics:

Equivalence relation. To recover a well-behaved structure (a cartesian-closed category), a coarser equivalence relation is necessary to satisfy the usual categorical laws of products and arrows. Indeed, composition by some copycat-like strategies might tamper with copy indices and result in a similar strategy, which only differs by the choice of copy indices.

Uniformity. If Player is allowed to play many times the same move, so must be Opponent. Strategies should be receptive to these copies but should not act differently depending on the index chosen by Opponent. Strategies should behave uniformly with respect to negative copy indices.

This chapter introduces a compositional framework to solve those problems by adjoining to event structures a notion of symmetry which can express that two events are essentially the same (ie. correspond to same move). This symmetry is represented as a proof-relevant equivalence relation permitting to mark the different copies as "symmetric".

Related work. The idea of adjoining copy indices to moves in order to represent nonlinear behaviours was first carried out in **[AJM00]**. The necessity in our framework to equip games with two symmetries, one positive and one negative, is reminiscent of Melliès' definition of uniformity in asynchronous games by bi-invariance under the action of two groups, one for Player reindexings, and one for Player reindexings **[Mel03]**.

Outline of the chapter. Section 1 introduces the expansion process turning a game A, with a specific shape (called an arena) to a game $!A$ where moves are

duplicated. We provide a few examples of non-linear programs and their representation in this setting and show that strategies on these expanded games support a coarser equivalence relation allowing for a richer equational theory.

Section 2 introduces a generalization of event structures to support this richer equivalence relation, called *event structures with symmetry* [**Win07**].

Section 3 exploits this new metalanguage and generalizes games and pre-strategies to $\sim$-games and pre-$\sim$-strategies satisfying uniformity, in a setting enriched with symmetry.

Section 4 explores how to compose these uniform pre-strategies and builds a category of $\sim$-strategies up to a generalization of isomorphism.

Section 5 defines a new equivalence relation on pre-$\sim$-strategies that allows us to identify strategies up the symmetry given by the $\sim$-games. For this relation to be congruence, $\sim$-games need to be strengthened to *thin concurrent games*.

Finally Section 6 proves that the resulting compositional structure forms a compact-closed category when restricting pre-$\sim$-strategies to $\sim$-strategies.

Contribution of the chapter. The construction of the compact-closed category in the setting without essential events is joint work with Pierre Clairambault and Glynn Winskel [**CCW15, CCW14**]. This chapter presents a generalization of this construction to the framework using essential events, introduced in Chapter 2.

1. Expanded arenas

In usual game semantics, the base category (of simple games) is also linear, i.e. naturally equipped with a symmetric monoidal closed structure, yielding a model of the linear λ-calculus. Two main solutions to overcome linearity arose. In the HO approach, types are represented as arenas and strategies as sets of non-linear plays (ie. plays where the same move from the arena can occur several times). In the AJM approach, a $\star$-autonomous category of simple games and linear strategies equipped with an equivalence relation is first constructed. This category supports an exponential comonad whose Kleisli category is cartesian-closed.

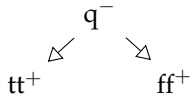
Our approach shares similarities with both methods. We first build a compact-closed category $\sim$-tCG$^{\cong}_{\circledcirc}$ whose equational theory is rich enough to support an exponential comonad and recover cartesian-closed categories. However, we will show (in the next chapter), how to isolate a cartesian-closed category inside $\sim$-tCG$^{\cong}_{\circledcirc}$ that is closer in spirit to HO games: nonlinear plays on an arena A will be represented by linear plays on a derived game $!A$.

In this thesis, types will always be interpreted by certain games, *arenas*:

DEFINITION 3.1. An **arena** is a *countable* game satisfying

- *Conflict-free.* All finite sets are consistent.
- *Forest.* If $a_1, a_2 \leq a \in A$, then either $a_1 \leq a_2$ or $a_2 \leq a_1$.
- *Alternation.* If $a_1 \rightarrowtriangle a_2$, then $\mathrm{pol}(a_1) \neq \mathrm{pol}(a_2)$.

For instance, the arena $\mathbb{B}$ for booleans is as follows:



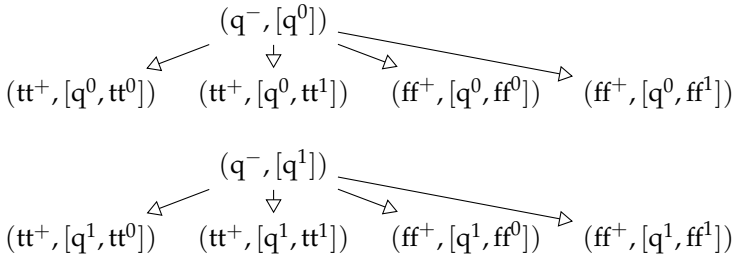
Unlike in Chapter 2, there is no conflict between the two positive moves. This implies that a strategy on this game can answer both true and false. The meaning of such strategies will become clearer in Chapter 5, Section 1.1.3.

Nonlinear plays. As mentioned earlier, we can represent a nonlinear strategy on an arena A as a strategy (as introduced in Chapter 2) on a derived arena $!A$. However $!(\cdot)$ will not be treated as an exponential modality (as in AJM games for instance). In $!A$, moves are duplicated deeply in the arena tree, to permit playing a certain move several times deep in the tree without having to play the events below it several times as well. This deep duplication is represented using *index functions*. An index function in an arena A is a pair $(a \in A, \alpha : [a] \to \mathbb{N})$. The function α gives copy indices for every move in the causal history of a (including a itself). Index functions naturally form an arena:

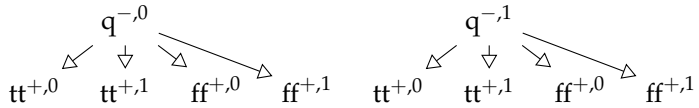
DEFINITION 3.2. Let A be an arena. Define the arena $!A$ as follows:

- *Events*: index functions (a, α) on A
- *Causality*: $(a, \alpha) \leq_{!A} (a', \alpha')$ whenever $a \leq_A a'$ and $\alpha' \restriction_{[a]} = \alpha$.
- *Polarities*: $\mathrm{pol}_{!A}(a, \alpha) = \mathrm{pol}_A(a)$.

The arena $!\mathbb{B}$ is (with $\mathbb{N} = \{0, 1\}$ for the picture):



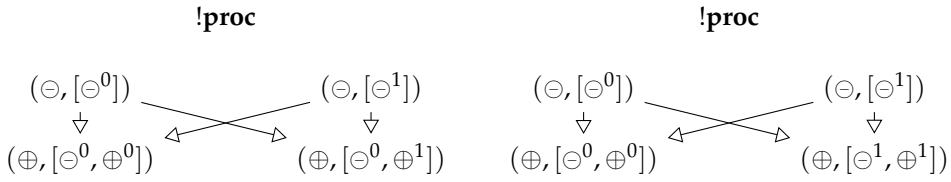
The index functions are written as lists $[a^{\alpha(a)}, b^{\alpha(b)}, \dots]$. This representation is cumbersome but also redundant. Indeed, if $(q^-, [q^0]) \twoheadrightarrow (tt^+, \alpha)$, we know automatically that $\alpha(q) = 0$ by definition of the causality in $!\mathbb{B}$. More generally, if we know the causal history of a $(a, \alpha) \in A$, the only extra information we need about α is $\alpha(a)$ as the rest of α can be looked up in the causal history. Since our diagrams always make causal histories of events explicit, we only need to represent the copy index of the top element of an index function. Hence this is how we draw $!\mathbb{B}$:



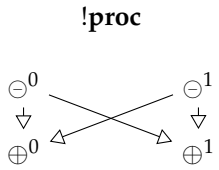
This is justified by the fact that an event $(a, \alpha) \in !A$ is uniquely determined by $(a, \alpha(a), \mathrm{just}(a, \alpha))$ where a is called the **label** of (a, α) (written $\mathrm{lbl}(a, \alpha)$), $\alpha(a)$ the **index** of (a, α) (written $\mathrm{ind}(a, \alpha)$) and $\mathrm{just}(a, \alpha)$ the **justifier** of (a, α) defined as the predecessor of (a, α) inside $!A$ if (a, α) is non-minimal ($\mathrm{just}(\cdot)$ is a partial function). Note that the predecessor must be unique because A (and $!A$) is a forest.

Representing a strategy requires to be more careful. Every (non-minimal) visible move s of a strategy $\sigma : S \rightharpoonup !A$ has a justifier $\mathrm{just}(s)$ which is defined as the unique move in $[s]$ whose image is $\mathrm{just}(\sigma s)$. Because strategies are not forests, the

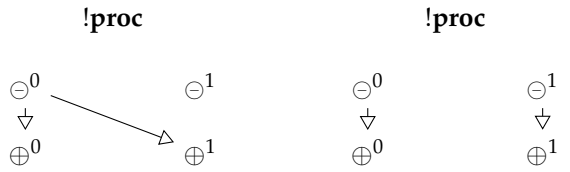
justifier might not be apparent from the causal history in the strategy. Consider the following two strategies on $!(\ominus \multimap \oplus)$ with explicit index functions:



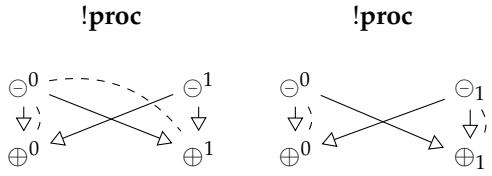
If we want to use the implicit representation of the move labels in $!A$, we end up with the same event structure:



The ambiguity here is on the projection on the game: is it the left or the right configuration of !**proc**?



To lift the ambiguity, we need to add more information about the implicit diagram to make it non-ambiguous, by indicating the causal history in the game by *pointer links* from a just(s) to s. This gives the following implicit representations:



Those pointer links have the same purpose as in HO game semantics: disambiguate which move justifies another move. The ambiguity caused in this example by concurrency also occurs in a (sequential) higher-order setting. In our setting, by courtesy, the justifier of a negative move is uniquely determined as its predecessor (for the causal order of the strategy). As a result, in the HO terminology, configurations of a strategy are analogous to P-views since Opponent always points to the preceding Player move.

Symmetries on $!A$. It is tempting to relax isomorphism of strategies to recover a richer equational theory. In our previous examples, we had complete freedom over the choice of copy indices for positive moves. It is crucial that this choice does not matter to recover a cartesian category (see Example 4.4).

Say an order-isomorphism $\theta : x \cong y$ between configurations of $!A$ is a **reindexing iso** when it preserves the label ($\mathrm{lbl} \circ \theta = \mathrm{lbl}$). It is **positive** when it preserves the copy index of negative moves (for all $\gamma^- \in x$, $\mathrm{ind} \circ \theta(\gamma) = \mathrm{ind}\,\gamma$) and **negative** when it preserves the copy index of positive moves.

Those reindexing isos induce a weaker equivalence relation on strategies. Two strategies $\sigma, \tau : !A$ are said to be **weakly isomorphic** when there exists an isomorphism $\varphi : S \cong T$ such that for all configuration $x \in \mathscr{C}(S)$, the bijection $\sigma x \simeq \tau(\varphi x)$ (induced by local injectivity) is a positive reindexing iso. This equivalence allows us to capture strategies exactly up to the choice of copy index of positive events.

Firstly, we need to prove that weak isomorphism is a congruence. This requires understanding the abstract structure of $!A$, as a *thin concurrent game* and understand what expressive power can be granted to strategies while still preserving the fact that weak isomorphism is a congruence. Those investigations will come down to the construction of another compact-closed category $\sim$-$\mathrm{tCG}_{\circledcirc}^{\cong}$.

In the next section, we first introduce the new metalanguage on which $\sim$-$\mathrm{tCG}_{\circledcirc}^{\cong}$ is based: event structures are replaced by event structures *with symmetry* that axiomatize the structure of reindexing isos of $!A$.

2. Event structures with symmetry

2.1. Isomorphism families. We have seen that the game $!A$ is naturally equipped with families of bijections expressing the fact that some configurations are symmetric. These families can be abstracted as *isomorphism families*:

DEFINITION 3.3 (Isomorphism family). Let A be an event structure and $\tilde{A}$ be a set of bijections between configurations of A. The family $\tilde{A}$ is an **isomorphism family** on A if it satisfies the following properties:

- *(Groupoid)* The set $\tilde{A}$ contains all identity bijections, and is stable under composition and inverse.
- *(Restriction)* For every bijection $\theta : x \simeq y \in \tilde{A}$ and $x' \in \mathscr{C}(A)$ such that $x' \subseteq x$, then the restriction $\theta \restriction x'$ of θ to x' is in $\tilde{A}$. In particular, $\theta\, x' \in \mathscr{C}(A)$.
- *(Extension)* For every bijection $\theta : x \simeq y \in \tilde{A}$ and every extension $x \subseteq x' \in \mathscr{C}(A)$, there exists a (non-necessarily unique) $y \subseteq y' \in \mathscr{C}(A)$ and an extension $\theta \subseteq \theta'$ such that $\theta' : x' \simeq y' \in \tilde{A}$.

In this case the pair $\mathcal{A} = (A, \tilde{A})$ is called an **event structure with symmetry (ess)**. We will use $\mathcal{S}, \mathcal{T}, \mathcal{A}, \mathcal{B}, \ldots$ to range over event structures with symmetry.

An isomorphism family on an event structure with partial (resp. total) polarities A is an isomorphism family $\tilde{A}$ on the underlying event structure such that all bijections in $\tilde{A}$ preserve polarities. In that case the pair $(A, \tilde{A})$ is called an **event structure with symmetry and partial (resp. total) polarities**, abbreviated as simply as **essp** for the partial polarities case. The notation $\subseteq^+$ and $\subseteq^-$ are generalized to symmetries of an essp, regarded as their graphs, in a direct way: $\theta \subseteq^+ \theta'$ when $\theta \subseteq \theta'$ and $\theta' \setminus \theta$ only contains pairs of positive moves (and similarly for $\subseteq^-$). Essps will be key when defining strategies in this setting enriched with symmetry.

Event structures with symmetry were first introduced by Winskel **[Win07]** in a more abstract presentation based on spans of open maps. This abstract presentation will be used to lift the compact-closed structure of $\mathrm{CG}_{\circledcirc}$ to $\sim$-$\mathrm{tCG}_{\circledcirc}^{\cong}$ (see

Section 6), but we chose the present concrete presentation in terms of isomorphism families as it is more elementary.

The definition above does not explicitly mention that the bijections need to be order-isomorphisms. It is actually a consequence of the (Restriction) axiom:

LEMMA 3.4. *Let $\mathcal{A}$ be an ess and $\theta : x \simeq y \in \tilde{A}$. Then, θ is an order-isomorphism.*

PROOF. Let $s \leq s' \in x$. Applying the restriction axiom to θ^{-1} and the configuration $[\theta s'] \subseteq y$ entails that $\theta^{-1}[\theta s']$ is a configuration so it is in particular down-closed. As $s' \in \theta^{-1}[\theta s']$, it follows that $s \in \theta^{-1}[\theta s']$. This directly implies $\theta s \leq \theta s'$ as $\theta s \in [\theta s']$. □

Since $\theta : x \simeq y \in \tilde{A}$ is an order-isomorphism, we will denote it via $\theta : x \cong y$ to indicate that it preserves and reflects the (implicit, inherited from $\leq_A$) ordering on x and y. Instead of $\theta : x \cong y \in \tilde{A}$, we will also often use the more compact notation $\theta : x \cong_{\tilde{A}} y$; and we will refer to θ as **a symmetry** between x and y.

Given a bijection θ, write $\operatorname{dom}\theta$ and $\operatorname{codom}\theta$ for its domain and codomain respectively. The existence of a symmetry θ between two configurations x and y of A ensures that x and y have isomorphic pasts and bisimilar futures.

The axiom (Extension) is equivalent to its one-step counterpart:

LEMMA 3.5 (One-step extension). *Let A be an event structure and $\tilde{A}$ be a family of bijections. It satisfies the (Extension) axiom if and only if for all $\theta : x \cong_{\tilde{A}} y$ and extension $a \in A$ of x, there exists $a' \in A$ such that $\theta \cup \{(a, a')\} \in \tilde{A}$ (in particular a' extends y).*

PROOF. Straightforward by induction. □

The game $!A$ along with its reindexing isos defined in the previous section give rise to examples of event structures with symmetry.

PROPOSITION 3.6. *For an arena A, the sets $\widetilde{!A}$ of* reindexing isos, *$\widetilde{!A}_-$ of* negative reindexing isos, *and $\widetilde{!A}_+$ of* positive reindexing isos, *are isomorphism families on $!A$.*

PROOF. The (Groupoid) axiom is easy to check for these three families. The (Restriction) axiom follows from all the θ being order-isomorphisms.

We check the (Extension) axiom for the first family using Lemma 3.5. Let $\theta : x \cong_{\widetilde{!A}} y$ and $(a, \alpha : [a] \to \mathbb{N})$ be an extension of x. Recall from the discussion below Definition 3.2 that events in $!A$ are entirely determined by their label, their *justifier* (immediate causal dependency), and copy index. Define $\zeta = \operatorname{just}(a, \alpha)$. We set the extension of θ to be (α, β), where β is set to be the unique event of $!A$ with justifier $\theta(\zeta)$ (or $\star$ if (a, α) was minimal), label a and copy index some fresh k not reached in y yet. This yields $\theta \cup \{(\alpha, \beta)\}$ an order-isomorphism between configurations of $!A$, preserving labels.

If $\theta : x \cong_{\widetilde{!A}_+} y$ and $(a, \alpha : [a] \to \mathbb{N})$ is a positive extension of x, the same reasoning applies. If it is a negative extension, then the unique possible extension of θ is (α, β) where β has justifier $\theta(\operatorname{just}(a, \alpha))$ (again with the convention that $\theta(\star) = \star$) and copy index $\operatorname{ind}\alpha$. Such β cannot be in y already: indeed, its pre-image through θ would be an event with label a, justifier just α and copy index $\operatorname{ind}\alpha$ – so would be α, absurd since $\alpha \notin x$. The reasoning for $\widetilde{!A}_-$ is dual. □

Basic operations on event structures extend to isomorphism families:

DEFINITION 3.7. Let $\mathcal{A}$ and $\mathcal{B}$ be ess. We build their **simple parallel composition** as $(A \parallel B, \tilde{A} \parallel \tilde{B})$ where $\tilde{A} \parallel \tilde{B}$ is the set of bijections of the form $\theta_1 \parallel \theta_2 : x \parallel y \simeq x' \parallel y'$ where $x, x' \in \mathscr{C}(A), y, y' \in \mathscr{C}(B)$, $\theta_1 \in \tilde{A}, \theta_2 \in \tilde{B}$ and $\theta_1 \parallel \theta_2$ is defined as $(i, a) \mapsto (i, \theta_i(a))$.

If $\mathcal{A}$ is an essp, its **dual** $\mathcal{A}^\perp$ is the same event structure with symmetry with reversed polarity.

Projection is not always defined: the set of visible events needs to be closed under symmetry to ensure the symmetry can be lifted to the projection:

LEMMA 3.8. *Let $\mathcal{E}$ be an ess and $V \subseteq E$* closed under symmetry*, in the sense that for all $\theta : x \cong_{\tilde{E}} y$, for all $e \in V \cap x$, we have $\theta\, e \in V$ as well. Then, defining*

$$\tilde{E} \downarrow V = \{\theta : x \simeq y \mid x, y \in \mathscr{C}(E \downarrow V),\ \exists \theta \subseteq \theta' \in \tilde{E},\ \theta' : [x]_E \cong_{\tilde{E}} [y]_E\}$$

we have that $\tilde{E} \downarrow V$ is an isomorphism family, making $\mathcal{E} \downarrow V = (E \downarrow V, \tilde{E} \downarrow V)$ into an event structure with symmetry.

PROOF. As usual the axiom (Groupoid) is clear. In this proof we abbreviate $[x]_E$ to $[x]$ for $x \in \mathscr{C}(E \downarrow V)$ for clarity.

(Restriction) Let $\theta : x \simeq y \in \tilde{E} \downarrow V$, and $x_0 \in \mathscr{C}(E \downarrow V)$ such that $x_0 \subseteq x$. By definition, θ extends to $\theta' : [x] \cong_{\tilde{E}} [y]$. We have $[x_0] \subseteq [x]$. Therefore, by (Restriction) on $\tilde{E}$ we have $\theta'_0 \subseteq \theta'$ with $\theta'_0 : [x_0] \cong_{\tilde{E}} y'_0$. Since V is closed under symmetry, $\theta'_0 \cap V^2 : x_0 \simeq y'_0 \cap V$ is still a bijection, which by definition is in $\tilde{E} \downarrow V$. It is clear by construction that $\theta'_0 \cap V^2 \subseteq \theta$.

(Extension) Let $\theta : x \simeq y \in \tilde{E} \downarrow V$, and $x \subseteq x_0 \in \mathscr{C}(E \downarrow V)$. By definition there is $\theta \subseteq \theta' : [x] \cong_{\tilde{E}} [y]$. We have $[x] \subseteq [x_0]$, therefore by (Extension) for $\tilde{E}$ there is $\theta'_0 : [x_0] \cong_{\tilde{E}} y'_0$. Again since V is closed under symmetry, $\theta'_0 \cap V^2 : x_0 \simeq y'_0 \cap V$ is still a bijection. By definition it is in $\tilde{E} \downarrow V$, and contains θ. □

2.2. Maps of event structures with symmetry. In the setting without symmetry, *partial maps of event structures* played a central role, providing an adequate notion of labeling functions for strategies $\sigma : S \rightharpoonup A$. In our new setting with symmetry, we will also need to consider a corresponding notion of partial map.

DEFINITION 3.9. Let $\mathcal{A}, \mathcal{B}$ be event structures with symmetry. A partial map of event structures $f : A \rightharpoonup B$ **preserves symmetry** iff for all $\theta : x \cong_{\tilde{A}} y$,

(1) f is defined on $s \in x$ if and only if f is defined on θs,
(2) $f\theta = \{(fa, fa') \mid (a, a') \in \theta \ \&\ fa \text{ defined}\}$ is in $\tilde{B}$.

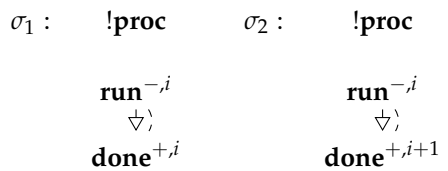
In that case, f is a **partial map of event structures with symmetry** , written $f : \mathcal{A} \rightharpoonup \mathcal{B}$. If $\mathcal{A}$ and $\mathcal{B}$ are essps, then f is a **map of event structures with symmetry and partial polarities** when it also preserves polarities.

Write $\mathcal{E}^\sim$ for the category of ess and *total* maps and $\mathcal{E}^\sim_\perp$ for the category of ess and *partial* maps. In $\mathcal{E}^\sim$ and $\mathcal{E}^\sim_\perp$, morphisms can be compared *up to symmetry*, abstracting away from the comparison of morphisms *up to the choice of copy indices* of the previous section.

DEFINITION 3.10. Let $\mathcal{B}$ be an event structure with symmetry and let $f, g : A \rightharpoonup B$ be maps of underlying event structures. They are **symmetric** (written $f \sim_{\tilde{B}} g$) when their domain coincide and for all $x \in \mathscr{C}(A)$, the bijection $\{(fs, gs) \mid s \in x \ \&\ fs \text{ defined}\}$ is in $\tilde{B}$.

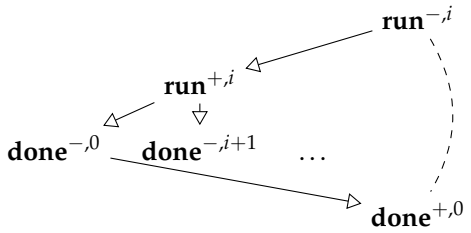
For this definition to make sense, A does not need to be equipped with an isomorphism family nor f, g need preserve it. This generality is useful to rephrase the definition of *weak isomorphism* given in Section 1. Two strategies σ, τ are weakly isomorphic iff there exists $\varphi : S \cong T$ such that $\tau \circ \varphi \sim_{\widetilde{!A_+}} \sigma$. This is however, not the end of the story: this equivalence is not a congruence.

EXAMPLE 3.11 (Uniformity problem). Let σ_1 and σ_2 be two strategies on !**proc**:



Because there is an obvious isomorphism sending $\mathbf{run}^{-,i}$ to $\mathbf{run}^{-,i}$ and $\mathbf{done}^{+,i}$ to $\mathbf{done}^{+,i+1}$, σ_1 and σ_2 are weakly isomorphic.

Consider now the strategy τ playing on $(!\mathbf{proc})^\perp \parallel !\mathbf{proc}$:



The strategy τ evaluates its argument and returns to toplevel only if its argument returned with a copy index of zero (it diverges otherwise). As a result $\tau \circledcirc \sigma_1$ contains a positive move whereas $\tau \circledcirc \sigma_2$ does not: they cannot be weakly isomorphic.

To solve this problem, we need only to consider those strategies that are blind to Opponent's choice of indices. This restriction is introduced in the next section.

3. Games with symmetry and uniform strategies

In this section, we show how to define a notion of uniform strategies. Forcing strategies to be uniform amounts to forcing negative events that only differ by their copy indices to have bisimilar futures. Symmetry is the right tool to ensure this property: we should force those negative events to be symmetric in the strategy. This means endowing strategies with a symmetry. Naturally, from partial maps of event structures $S \rightharpoonup A$ strategies should become partial maps of event structures with symmetry $\mathcal{S} \rightharpoonup \mathcal{A}$.

First, we define the generalized notion of games in a setting with symmetry:

DEFINITION 3.12. A **$\sim$-game** is an essp $\mathcal{A}$ such that for any $\theta : x \cong_{\tilde{A}} y$, and for any extensions $\theta \subseteq^+ \theta_1 : x_1 \cong_{\tilde{A}} y_1$ and $\theta \subseteq^- \theta_2 : x_2 \cong_{\tilde{A}} y_2$, then θ_1 and θ_2 are compatible: $\theta_1 \cup \theta_2 \in \tilde{A}$.

This extra condition is a generalization of race-freeness to isomorphism families. In particular, if $\mathcal{A}$ is a $\sim$-game, then A is a race-free game by letting θ, θ_1 and θ_2 be identities. This generalization is necessary to ensure that copycat fits in this new framework – see Section 3.2. These $\sim$-games naturally supports parallel composition and the dual operation.

In the case of $!A$, it is not clear which isomorphism family (among $\widetilde{!A}$, $\widetilde{!A_+}$, $\widetilde{!A_-}$) to use as the codomain. Since we want $\mathcal{S}$ to contain all the negative symmetries (to ensure that negative symmetric moves have similar futures). It is natural to ask that strategies should be certain maps of event structures with symmetry $\sigma : \mathcal{S} \to (!A, \widetilde{!A_-})$. It is however not expressive enough:

EXAMPLE 3.13. Consider the following strategy on $!\mathbf{proc}$:

$$\begin{array}{ccc} \mathbf{run}^{-,0} & \mathbf{run}^{-,1} & \cdots \\ \downarrow & \downarrow & \\ \mathbf{done}^{+,0} & \mathbf{done}^{+,1} & \cdots \end{array}$$

Writing S for this event structure, there is no isomorphism family $\tilde{S}$ containing a symmetry $\{\mathbf{run}_0\} \cong_{\tilde{S}} \{\mathbf{run}_1\}$ such that the obvious labelling defines a map of event structures with symmetry $\sigma : (S, \tilde{S}) \to \mathcal{A}$ where $\mathcal{A} = (!\mathbf{proc}, \widetilde{(!\mathbf{proc})_-})$. Indeed, by the extension axiom we would need to have $\theta : \{\mathbf{run}_0, \mathbf{done}_0\} \cong_{\tilde{S}} \{\mathbf{run}_1, \mathbf{done}_1\}$. Note that $\sigma\theta \notin \widetilde{(!\mathbf{proc})_-}$ since $\sigma\theta$ does not preserve the copy index of the positive move $\mathbf{done}_0$. However $\sigma\theta \in \widetilde{!\mathbf{proc}}$.

3.1. Uniform pre-strategies. We now move on to defining (uncovered) strategies in this new setting as certain maps of event structures with symmetry. As in Chapter 2, we first investigate what minimal structure is necessary for composition, leading to a notion of pre-$\sim$-strategies. Then, $\sim$-strategies arise as those pre-$\sim$-strategies that are invariant under composition with copycat. Remember that a map of event structures with symmetry $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ to a $\sim$-game $\mathcal{A}$ induces a choice of partial polarity that makes $\mathcal{S}$ an event structure with symmetry and partial polarities. Remember also that, in this case, an event $s \in S$ is nonnegative when it is positive (ie. σs is defined and positive) or neutral (ie. σs undefined).

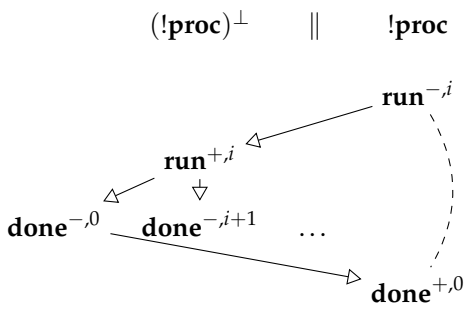
3.1.1. *$\sim$-receptivity.* First, as long as strategies are mere maps of event structures with symmetry, nothing prevents the isomorphism family from being reduced to identities. To force the symmetry on a strategy to be non-trivial, we introduce the condition of $\sim$-receptivity:

DEFINITION 3.14. If $\mathcal{A}$ is a $\sim$-game, a partial map of event structures with symmetry $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ is **$\sim$-receptive** iff for all $\theta : x_1 \cong_{\tilde{S}} x_2$, for all negative extensions s_1 of x_1, and a_2 of σx_2 such that $\sigma\theta \cup \{(\sigma s_1, a_2)\} \in \tilde{A}$, there is a unique s_2 such that $\sigma s_2 = a_2$, and $\theta \cup \{(s_1, s_2)\} \in \tilde{S}$.

In the previous example, this condition forces $\{\mathbf{run}_0\} \cong_{\tilde{S}} \{\mathbf{run}_1\}$. It is also important for the pullback of pre-$\sim$-strategies to exist (See Section 4.1). Remark that $\sim$-receptivity does not imply receptivity.

This condition indeed rules out non-uniform behaviours:

EXAMPLE 3.15. Remember the strategy $\sigma : !\mathbf{proc} \multimap !\mathbf{proc}$ of Example 3.11:



There is no symmetry $\tilde{S}$ on S that makes σ into a $\sim$-receptive map of event structures with symmetry $(S, \tilde{S}) \to !\mathbf{proc}^{\perp} \parallel !\mathbf{proc}$. Applied to the identity bijection on $\{(0, \mathbf{run}^{-,i}), (1, \mathbf{run}^{+,i})\}$, $\sim$-receptivity would entail that

$$\theta : \{(0, \mathbf{run}^{-,i}), (1, \mathbf{run}^{+,i}), (1, \mathbf{done}^{-,0})\} \cong_{\tilde{S}} \{(0, \mathbf{run}^{-,i}), (1, \mathbf{run}^{+,i}), (1, \mathbf{done}^{-,1})\}.$$

By the extension property, there would be an event s of S such that θ extends to:

$$\{(0, \mathbf{run}^{-,i}), (1, \mathbf{run}^{+,i}), (1, \mathbf{done}^{-,0}), (0, \mathbf{done}^{+,0})\}$$
$$\cong_{\tilde{S}} \{(0, \mathbf{run}^{-,i}), (1, \mathbf{run}^{+,i}), (1, \mathbf{done}^{-,1}), s\}.$$

In particular $(1, \mathbf{done}^{-,1}) \rightarrowtriangle s$, which is absurd as there is no such event in S.

Before we go on, let us mention in passing the following lemma: to check $\sim$-receptivity for a map, it is enough to look at extensions of identity symmetries.

LEMMA 3.16. *Let $\mathcal{A}$ be a tcg and $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ be a map of ess. Then, σ is $\sim$-receptive iff for all $x \in \mathscr{C}(S)$ and s_1 a negative extension of x, for all $\mathrm{id}_{\sigma x} \cup \{(\sigma s_1, a_2)\} \in \tilde{A}$, there exists a unique s_2 such that $\sigma s_2 = a_2$, and we have $\mathrm{id}_x \cup \{(s_1, s_2)\} \in \tilde{S}$.*

PROOF. *only if.* Particular case of the definition of $\sim$-receptivity.

if. Assume $\theta : x_1 \cong_{\tilde{S}} x_2$, with x_1 extended by a negative s_1, and σx_2 extended by a negative a_2, such that $\sigma\theta \cup \{(\sigma s_1, a_2)\} \in \tilde{A}$. By (Extension), there is an extension of the form (s_1, s_1') of θ (with $s_1' \in S$). Since σ is a map of ess, we must have $\sigma\theta \cup \{(\sigma s_1, \sigma s_1')\} \in \tilde{A}$ as well. By (Groupoid), it follows that $\mathrm{id}_{\sigma x_2} \cup \{(\sigma s_1', a_2)\} \in \tilde{A}$. By hypothesis, we get a unique s_2 such that $\sigma s_2 = a_2$, satisfying $\mathrm{id}_{x_2} \cup \{(s_1', s_2)\} \in \tilde{S}$. Finally, by (Groupoid) again, $\theta \cup \{(s_1, s_2)\} \in \tilde{S}$. □

3.1.2. *Thin.* To make sure that essential events are closed under symmetry (for the hiding to be well-defined) we ask that symmetries of strategies should be well-behaved: nonnegative extensions should be unique.[α]

DEFINITION 3.17 (Thin symmetry). If $\mathcal{A}$ is a $\sim$-game, for $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ a partial map, then the following conditions are equivalent:

(i) For all $\theta : x \cong_{\tilde{S}} y$ such that $x \overset{s}{\text{—}\!\subset} x'$ for a nonnegative $s \in S$, there is a **unique** $s' \in S$ such that $\theta \cup \{(s, s')\} \in \tilde{S}$.
(ii) Let x be a configuration of S that can be extended by a nonnegative $s \in S$. Then, for all $\theta \in \tilde{S}$ whose domain is $x \cup \{s\}$ and which contains id_x, we have that $\theta = \mathrm{id}_{x \cup \{s\}}$.

[α] In [**CCW14**], the problem with essential events does not occur, as there are only *covered* strategies, and the thin condition is only required later to ensure that weak isomorphism is a congruence.

Then we say $\mathcal{S}$ (and σ) is **thin** .

PROOF. The condition (ii) is just a special case of (i). We prove (i) assuming (ii). Let $\theta : x \cong_{\tilde{S}} y$ and a positive extension s to x. Assume there are two extensions of θ to $x' = x \cup \{s\}$: $\theta_1 : x' \cong_{\tilde{S}} y'_1$ and $\theta_2 : x' \cong_{\tilde{S}} y'_2$. Since both contain θ, $\theta_2 \circ \theta_1^{-1}$ extends id_y by a positive $(\theta_1(s), \theta_2(s))$. By (ii) we have $\theta_1(s) = \theta_2(s)$, hence $\theta_1 = \theta_2$ follows as desired. □

Towards composition, we check that the set of essential events of a pre-$\sim$-strategy is indeed closed under symmetry ($\sim$-receptivity is not necessary for this result to hold):

LEMMA 3.18. *Let $\mathcal{A}$ be a $\sim$-game, and $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ a map of event structures with symmetry such that $\mathcal{S}$ is thin. The set of essential events of S is closed under symmetry.*

PROOF. Let $s_1 \in S$ be an essential event and $x \in \mathscr{C}(S)$ with incompatible extensions s_1 and s_2. Consider $\theta \in \tilde{S}$ defined at s_1. Write θ_x for the restriction of θ to x. Since x extends by s_2, then there exists $s'_2 \in S$, extension of θx and $\theta_x \cup \{(s_2, s'_2)\} \in \tilde{S}$. If $\theta x \cup \{\theta s_1, s'_2\}$ is not a configuration of S, then this proves that θs_1 is indeed an essential event.

Otherwise, $(\theta_x \cup \{s_2, s'_2\})^{-1}$ can extend via $(\theta s, s'_1)$ for some s'_1 and:

$$x \cup \{s'_1\} \cong \theta x \cup \{\theta s_1\} \cong x \cup \{s_1\}$$

This particular isomorphism is the extension of the identity on x by the non-negative pair (s'_1, s_1) hence by thin $s_1 = s'_1$. But this means that $x \cup \{s_1, s_2\} \in \mathscr{C}(S)$ which is absurd. □

3.1.3. *Pre-$\sim$-strategies.* We can now generalize uncovered pre-strategies:

DEFINITION 3.19 (Pre-$\sim$-strategy). A **pre-$\sim$-strategy** on a $\sim$-game $\mathcal{A}$ is a $\sim$-receptive partial map of event structures with symmetry $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ such that $\mathcal{S}$ is thin. Similarly a **pre-$\sim$-strategy** from tcg $\mathcal{A}$ to tcg $\mathcal{B}$ is a $\sim$-receptive map of event structures with symmetry $\mathcal{S} \rightharpoonup \mathcal{A}^\perp \parallel \mathcal{B}$ such that $\mathcal{S}$ is thin.

Note that we dropped the uncovered attribute: strategies will be uncovered by default. Any pre-$\sim$-strategy $\mathcal{S} \rightharpoonup \mathcal{A}$ induces a (uncovered) pre-strategy $S \rightharpoonup A$ by simply forgetting the symmetry. Note that $\sim$-receptivity only implies the uniqueness part of receptivity, not the existence. If σ is both receptive and $\sim$-receptive, we say it is **strongly receptive**.

As a result, any pre-$\sim$-strategy $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ induces a visible part $\sigma_\downarrow : \mathcal{S} \downarrow V \to \mathcal{A}$ where V is the set of events sent to A, via Lemma 3.8.

3.2. Copycat. As a key example, we show how the copycat strategy can be equipped with a symmetry, and made into a pre-$\sim$-strategy. Recall that the copycat strategy on game A:

$$\mathfrak{c}_A : \mathfrak{CC}_A \to A^\perp \parallel A$$

where $\mathfrak{CC}_A$ has the same events as $A^\perp \parallel A$, but additional immediate causal links from negative events on one side to their positive counterpart on the other. Consequently, configurations $x \in \mathscr{C}(\mathfrak{CC}_A)$ decompose as $x = x_1 \parallel x_2 \in \mathscr{C}(A^\perp \parallel A)$.

The following definition is forced by the requirement that the map $\mathfrak{c}_A$ should be a map of ess, and that each symmetry should be an order-iso.

DEFINITION 3.20. Let $\mathcal{A}$ be a tcg. Given $x = x_1 \parallel x_2 \in \mathscr{C}(\mathbb{C}_A)$, $y = y_1 \parallel y_2 \in \mathscr{C}(\mathbb{C}_A)$, the set of *symmetries* between x and y (written $\mathbb{C}_{\tilde{A}}$) comprises any bijection $\theta = \theta_1 \parallel \theta_2$ such that $\theta_1, \theta_2 \in \tilde{A}$, and which is an order-iso (for the order on x, y induced by $\leq_{\mathbb{C}_A}$).

This intuitive definition makes sense. However in order to reason on such symmetries, it will be convenient to rely on a more high-level characterization that does not explicitly require an order-isomorphism. To introduce it, recall first from Chapter 2 that configurations $x \in \mathscr{C}(\mathbb{C}_A)$ are exactly those $x_1 \parallel x_2 \in \mathscr{C}(A \parallel A)$ such that (with polarity in A):

$$x_2 \supseteq^- x_1 \cap x_2 \subseteq^+ x_1$$

Furthermore, this relation between x_2 and x_1 is a partial order called the "Scott order", written $x_2 \sqsubseteq_A x_1$. If $\mathcal{A}$ is a tcg, we now observe the following.

PROPOSITION 3.21. *The set $\mathbb{C}_{\tilde{A}}$ is equivalently defined as comprising the bijections*

$$\theta_1 \parallel \theta_2 : x_1 \parallel x_2 \simeq_{\tilde{A}^\perp \parallel \tilde{A}} y_1 \parallel y_2$$

satisfying the further condition that for all $a \in x_1 \cap x_2$, we have $\theta_1(a) = \theta_2(a)$.

In other words, $\mathbb{C}_{\tilde{A}}$ comprises the bijections $\theta_1 \parallel \theta_2 \in \tilde{A}^\perp \parallel \tilde{A}$ such that $\theta_2 \supseteq^- \theta_1 \cap \theta_2 \subseteq^+ \theta_1$, i.e.

$$\theta_2 \sqsubseteq_{\tilde{A}} \theta_1$$

This justifies the notation $\mathbb{C}_{\tilde{A}}$, as this agrees with the description of configurations of copycat via the Scott order.

PROOF. Take $\theta = \theta_1 \parallel \theta_2 : x_1 \parallel x_2 \cong y_1 \parallel y_2$.

If θ is an order-isomorphism, then take $a \in x_1 \cap x_2$. Assume without loss of generality that $\mathrm{pol}_A(a) = +$, so that we have $(0, a) \rightarrowtriangle (1, a)$ in $\mathbb{C}_A$. But then since θ is an order-iso, it preserves immediate causal dependency, therefore $(0, \theta_1\, a) \rightarrowtriangle (1, \theta_2\, a)$. But since these two events are in different components of $A^\perp \parallel A$, this necessarily means that $\theta_1\, a = \theta_2\, a$ as required (using Lemma 2.10).

Conversely, assume that for all $a \in x_1 \cap x_2, \theta_1\, a = \theta_2\, a$. Using again Lemma 2.10, it is immediate that θ preserves immediate causal links. The same reasoning applies to θ^{-1} (it is easy to show that the hypothesis is stable under inverse), so it reflects immediate causal links as well; and is an order-iso. □

We now check the axioms for isomorphism families.

LEMMA 3.22. *The family $\mathbb{C}_{\tilde{A}}$ satisfies the axioms (Groupoid) and (Restriction) of isomorphism families.*

PROOF. Immediate from Definition 3.20. □

We prove the extension axiom separately: it relies on the extra axiom of $\sim$-games (see **[CCW14]** for a counterexample without this condition):

PROPOSITION 3.23. *Let $\mathcal{A}$ be a tcg. Then, writing $\mathbb{C}_{\mathcal{A}} = (\mathbb{C}_A, \mathbb{C}_{\tilde{A}})$, the map*

$$\varpi_A : \mathbb{C}_{\mathcal{A}} \to \mathcal{A}^\perp \parallel \mathcal{A}$$

is a pre-$\sim$-strategy.

PROOF. **$\mathbb{C}_{\tilde{A}}$ is an isomorphism family.** By Lemma 3.22 we already know that $\mathbb{C}_{\tilde{A}}$ satisfies all axioms for an isomorphism family except for (Extension), which we establish now.

Let $\theta_1 \parallel \theta_2 : x \parallel y \cong_{\mathbb{C}_{\tilde{A}}} x' \parallel y'$. Assume *e.g.* $x \parallel y \overset{(2,a)}{\text{—}\!\subset}$. There are two cases:

- If $\mathrm{pol}_A(a) = -$, then by (Extension) for $\tilde{A}^\perp \parallel \tilde{A}$ we have $\theta_1 \parallel \theta_2 \subseteq \theta_1 \parallel \theta_2' \in \tilde{A}^\perp \parallel \tilde{A}$ whose domain is $x \parallel y \cup \{a\}$. Its codomain is $x' \parallel y' \cup \{a'\}$. Since $\mathrm{pol}_A(a) = -$, we cannot have $a' \in x'$ – indeed $x' \supseteq^+ x' \cap y' \subseteq^- y'$, so we would have $a' \in y'$ as well, absurd. So we have $x' \cap (y' \cup \{a'\}) = x' \cap y' \subseteq^+ x'$, and $x' \cap (y' \cup \{a'\}) = x' \cap y' \subseteq^- y' \subseteq^- y' \cup \{a'\}$, which establishes that $x' \parallel (y' \cup \{a'\}) \in \mathscr{C}(\mathbb{C}_A)$.

  Likewise we have $\theta_1 \cap \theta_2' = \theta_1 \cap \theta_2$, hence we still have $\theta_1 \cap \theta_2' \subseteq^+ \theta_1$ but also $\theta_1 \cap \theta_2' \subseteq^- \theta_2 \subseteq^- \theta_2'$, therefore $\theta_1 \parallel \theta_2' \in \mathbb{C}_{\tilde{A}}$.
- If $\mathrm{pol}_A(a) = +$ is positive then $a \in x$ as well. Thus, $[a] \subseteq x \cap y$. Therefore, we have $(x \cap y) \cup \{a\} \in \mathscr{C}(A)$, and $(x \cap y) \cup \{a\} \subseteq x$. Define $\theta_1' = \theta_1 \upharpoonright (x \cap y) \cup \{a\}$. We have:

  $$\theta_1' \supseteq^+ \theta_1 \cap \theta_2 \subseteq^- \theta_2$$

  By construction, the domains of θ_1' (which is $(x \cap y) \cup \{a\}$) and the domain of θ_2 (which is y) are compatible, so by definition of $\sim$-games, $\theta_2' = \theta_1' \cup \theta_2 \in \tilde{A}$, and by construction its domain is $y \cup \{a\}$. To sum up, we have:

  $$\theta_1 \supseteq^+ \theta_1 \cap \theta_2' \subseteq^- \theta_2'$$

  Hence $\theta_1 \parallel \theta_2' \in \mathbb{C}_{\tilde{A}}$ provides the required extension.

ϖ_A is a pre-$\sim$-strategy. It is obvious that $\varpi_A : \mathbb{C}_{\mathcal{A}} \to \mathcal{A}^\perp \parallel \mathcal{A}$ preserves symmetry. It remains finally to show that it is $\sim$-receptive, for which we apply Lemma 3.16.

$\sim$-receptivity: Assume $x \parallel y \in \mathscr{C}(\mathbb{C}_A)$ can be extended for instance by $(2, a^-)$ in $\mathbb{C}_A$ and by $(2, b^-)$ in $A^\perp \parallel A$, such that:

$$\mathrm{id}_x \parallel (\mathrm{id}_y \cup \{(a,b)\}) \in \tilde{A}^\perp \parallel \tilde{A}$$

We need to check that this is a valid extension in $\mathbb{C}_{\tilde{A}}$ as well. By the characterization of Proposition 3.21, we only have to check that $\mathrm{id}_x\ c = (\mathrm{id}_y \cup \{(a,b)\})\ c$ for each $c \in x \cap (y \cup \{a\})$, but in fact we must have $c \in x \cap y$. Indeed, we cannot have $a \in x$, as by $x \supseteq^+ x \cap y \subseteq^- y$ and $\mathrm{pol}_A(a) = -$ that would imply $a \in y$ as well, absurd. So the verification is obvious.

Thin. Assume we have $\mathrm{id}_{x \parallel y}$ that can be extended by a positive (e, e') to θ in $\mathbb{C}_{\tilde{A}}$. For instance $e = (1, a)$ and $e' = (1, a')$ (right component). By construction of copycat, since $(0, a) \rightarrowtriangle (1, a)$ we must have $\theta(0, a) \rightarrowtriangle \theta(1, a)$. Since $\theta(0, a) = (0, a)$ and $\theta(0, a') = (1, a')$ it follows that $(0, a) \rightarrowtriangle (1, a')$ hence $a = a'$ as desired. □

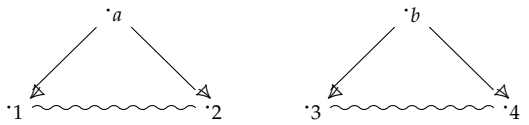
Having defined games and pre-strategies in this new metalanguage of event structures with symmetry, we now look at how composition should be updated.

4. Composition of uniform strategies

In this section, we investigate how to generalize the definition of composition from Chapter 2 to uniform strategies. The difficulty is to build an isomorphism family on top of $T \circledcirc S$, which amounts to show that uniform strategies compose.

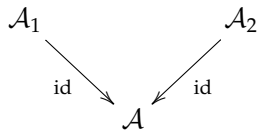
4.1. Interaction of pre-$\sim$-strategies. Remember that, in the covered case, interaction is given by pullback. However, the category $\mathcal{E}^{\sim}$ of ess and total maps does not have pullbacks in general:

EXAMPLE 3.24. Consider A the following event structure:



Write $\mathcal{A}$ for A equipped with the maximal isomorphism family: all order-isomorphisms are in the family. Write $\mathcal{A}_1$ for the sub-event structure with symmetry where $\cdot_1$ can only be sent to itself and to $\cdot_3$; and $\cdot_2$ can only be sent to itself and to $\cdot_4$. Similarly, write $\mathcal{A}_2$ for that where $\cdot_1$ can only be sent to $\cdot_4$ and $\cdot_2$ to $\cdot_3$.

We have the following diagram:



where id is the identity on events but not the identity map in $\mathcal{E}^{\sim}$, since the isomorphism families are distinct.

Assume this diagram has a pullback $(\mathcal{A}_3, \Pi_1, \Pi_2)$. Since the projection functor $(A, \tilde{A}) \mapsto A : \mathcal{E}^{\sim} \to \mathcal{E}$ has a left adjoint (the functor taking A to $(A, \{\mathrm{id}_x \mid x \in \mathscr{C}(A)\})$), it preserves pullbacks up to isomorphism. So the underlying event structure of $\mathcal{A}_3$ is A and the projection maps are both identities on events. The isomorphism $\{(\cdot_a, \cdot_b)\} : \{\cdot_a\} \cong_{\tilde{A}_3} \{\cdot_b\}$ must be in $\tilde{A}_3$ as it is in both $\tilde{A}_1$ and $\tilde{A}_2$. However, its left-hand side $\{\cdot_a\}$ can be extended with $\cdot_1$, so by the extension property we have:

$$\{(\cdot_a, \cdot_b), (\cdot_1, \cdot_i)\} : \{\cdot_a, \cdot_1\} \cong_{\tilde{A}_3} \{\cdot_b, \cdot_i\}$$

with $i \in \{3, 4\}$. But by construction such an isomorphism cannot be in both $\tilde{A}_1$ and $\tilde{A}_2$, absurd.

The problem is that the two isomorphism families do not agree on the extension. This will not happen if the maps are $\sim$-receptive and have dual codomains: any extension will be negative for one of the two strategies involved. This pre-$\sim$-strategy will be receptive to any extension the other one chooses, by $\sim$-receptivity.

Consider two pre-$\sim$-strategies $\sigma : S \rightharpoonup \mathcal{A}$ and $\tau : \mathcal{T} \rightharpoonup \mathcal{A}^{\perp}$ on dual games. If $\theta : w \simeq z$ with $w, z \in \mathscr{C}(S \wedge T)$ is such that

$(*)$ $\quad \Pi_1 p$ is defined iff $\Pi_1(\theta p)$ is defined for $p \in w$ (and similarly for Π_2)

then $\theta_S := \Pi_1\theta : \Pi_1 w \simeq \Pi_1 z$ and $\theta_T := \Pi_2\theta : \Pi_2 w \simeq \Pi_2 z$ are well-defined bijections. This remark allows us to define the symmetry on the interaction $S \wedge T$. On top of $S \wedge T$, consider the family $\tilde{S} \wedge \tilde{T}$ containing those bijections $\theta : w \cong z$ satisfying (*) such that $\theta_S : \Pi_1 w \simeq \Pi_1 z \in \tilde{S}$ and $\theta_T : \Pi_2 w \simeq \Pi_2 z \in \tilde{T}$. Bearing in mind the correspondence between configurations w of $S \wedge T$ and secured bijections of the shape

$$\Pi_1 w \parallel (\Pi_2 w)_* \simeq (\Pi_1 w)_\star \parallel \Pi_2 w.$$

In the following, such bijection will be abbreviated as

$$\Pi_1 w \xrightarrow{\ \sigma\ } (\sigma \wedge \tau) w \xleftarrow{\ \tau\ } \Pi_2 w \ .$$

There is an order-isomorphism between those bijections $\theta \in \tilde{S} \wedge \tilde{T}$ and commutative squares between the corresponding secured bijections (ordered by componentwise union):

$$\begin{array}{ccc} \Pi_1 w & \xrightarrow{\ \sigma\ } (\sigma \wedge \tau) w & \xleftarrow{\ \tau\ } \Pi_2 w \\ {\scriptstyle \theta_S}\wr\!\!\mid_{\tilde{S}} & & {\scriptstyle \theta_T}\wr\!\!\mid_{\tilde{T}} \\ \Pi_1 z & \xrightarrow{\ \sigma\ } (\sigma \wedge \tau) w & \xleftarrow{\ \tau\ } \Pi_2 z \end{array}$$

This construction satisfies a straightforward generalization of the universal property of Lemma 2.47. Remember that if we have two maps $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ and $\tau : \mathcal{T} \rightharpoonup \mathcal{B}$, a cone is a tuple $(\mathcal{X}, f : \mathcal{X} \rightharpoonup \mathcal{S}, g : \mathcal{X} \rightharpoonup \mathcal{T})$ such that the square commutes. A cone $(\mathcal{X}, f, g)$ **does not synchronize on neutral events** when for $s \in \mathcal{X}$ such that $f\,s$ and $g\,s$ are defined, then so is $\sigma(f\,s)$.

LEMMA 3.25. *Let $\mathcal{A}$ be an event structure with symmetry and* total *polarities (all events are either positive or negative). Let $\sigma : \mathcal{S} \to \mathcal{A}$ and $\tau : \mathcal{T} \to \mathcal{A}^\perp$ be $\sim$-receptive maps of ess. We have:*

(1) $\mathcal{S} \wedge \mathcal{T} = (S \wedge T, \tilde{S} \wedge \tilde{T})$ is an event structure with symmetry,
(2) the maps $\Pi_1 : \mathcal{S} \wedge \mathcal{T} \rightharpoonup S$ and $\Pi_2 : \mathcal{S} \wedge \mathcal{T} \rightharpoonup T$ preserve symmetry,
(3) the cone $(\mathcal{S} \wedge \mathcal{T}, \Pi_1, \Pi_2)$ is universal among cones $(\mathcal{X}, f, g)$ such that f and g do not synchronize on neutral events.

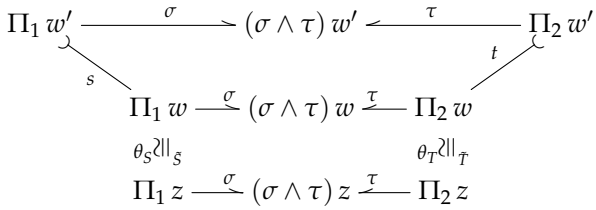
The proof of this lemma crucially relies on σ and τ playing on dual games.

PROOF. **(1)** The (Groupoid) and (Restriction) axioms are direct consequences of the corresponding conditions for $\tilde{S}$ and $\tilde{T}$. We check extension.

Let $\theta : w \cong_{\tilde{S} \wedge \tilde{T}} z$. Assume w can be extended by an event $p \in S \wedge T$ to w'.

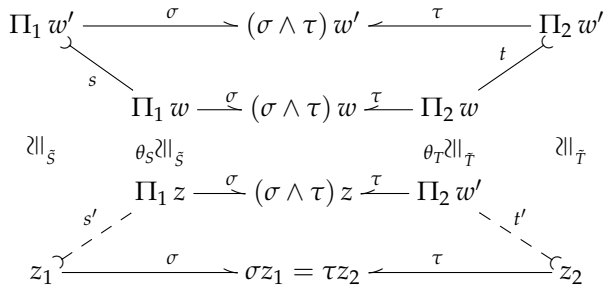
Extension by a neutral event. Assume $(\sigma \wedge \tau)p$ is not defined. As a result, either $\Pi_1 p$ is defined or $\Pi_2 p$ is. Assume $\Pi_1 p$ is. Then $\theta_S : \Pi_1 w \cong_{\tilde{S}} \Pi_1 z$ can extend by $(\Pi_1 p, s')$ by extension of $\tilde{S}$. Since s' is a neutral event for S, there exists an extension $p' \in S \wedge T$ of z and $\Pi_1 p' = s'$. As a result θ extends by (p, p') as desired.

Extension by a visible event. Assume $(\sigma \wedge \tau)p$ is defined. Since σ and τ have dual codomains, w.l.o.g, we can assume that $s := \Pi_1 p$ is positive and $t := \Pi_2 p$ is negative. We have the following picture:



We first use the extension property on θ_S as $\Pi_1 w \overset{s}{\hookrightarrow} : \theta_S$ extends by (s, s'). Since $\sigma\theta_S = \tau\theta_T$, this means that $\tau\theta_T$ extends by $(\sigma s, \sigma s')$ which is negative in $\mathcal{A}^\perp$. By $\sim$-receptivity of τ, it follows that θ_T extends by (t, t') with $\tau t = \sigma s$ and $\tau t' = \sigma s'$.

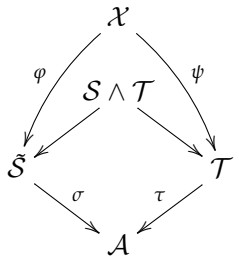
The picture is now:



The obtained bijection $\varphi : z_1 \simeq z_2$ is secured. As a result, w' extends by some $p' \in S \wedge T$ with $\Pi_1 p' = s'$ and $\Pi_2 p' = t'$, and θ extends by (p, p').

(2) Consequence of the definition of $\tilde{S} \wedge \tilde{T}$.

(3) Assume we have two morphisms of ess $\varphi : \mathcal{X} \rightharpoonup \mathcal{S}$ and $\psi : \mathcal{X} \rightharpoonup \mathcal{T}$ not synchronizing on neutral events, such that the square commutes:



Applying the universal property at the level of event structures (Lemma 2.47), there is a map of event structures $\langle \varphi, \psi \rangle : X \to S \wedge T$ making the two triangles commute, which is unique in $\mathcal{E}$. This uniqueness lifts to $\mathcal{E}_\perp^\sim$ as the forgetful functor $\mathcal{E}_\perp^\sim \to \mathcal{E}_\perp$ is faithful. To conclude we need only to prove that $\langle \varphi, \psi \rangle$ preserves symmetry and is thus a morphism in $\mathcal{E}_\perp^\sim$. An isomorphism $\theta : x \cong_{\bar{X}} y$ is transported to a bijection $\langle \varphi, \psi \rangle \theta : \langle \varphi, \psi \rangle x \simeq \langle \varphi, \psi \rangle y$ such that $(\langle \varphi, \psi \rangle \theta)_S = \varphi \theta$ and $(\langle \varphi, \psi \rangle \theta)_T = \psi \theta$, thus $\langle \varphi, \psi \rangle \theta \in \tilde{S} \wedge \tilde{T}$ by definition. □

We note in passing, that when both symmetries are thin, the resulting symmetry is trivial: none of the strategies can start to introduce non-trivial symmetry:

LEMMA 3.26. *Let $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ and $\tau : \mathcal{T} \rightharpoonup \mathcal{A}^\perp$ be (covered) pre-$\sim$-strategies such that $\mathcal{S}$ and $\mathcal{T}$ are thin. The isomorphism family $\tilde{S} \wedge \tilde{T}$ is trivial (reduced to identities).*

PROOF. We prove by induction that all bijections in $\tilde{S} \wedge \tilde{T}$ are identities. Let $z \in \mathscr{C}(S \wedge T)$ and assume id_z extends by (p, p') to $\theta \in \tilde{S} \wedge \tilde{T}$. Assume for instance $\Pi_2 e$ is nonnegative in T (the other case is similar). By construction $\mathrm{id}_{\Pi_2 z}$ extends to $\theta_T = \Pi_2 \theta \in \tilde{T}$ by positive events $(\Pi_2 p, \Pi_2 p')$, hence $\Pi_2 p = \Pi_2 p'$ and θ_T is the identity because $\tilde{T}$ is thin. By local injectivity of Π_2 it follows that p and p' must be equal, or incompatible extensions of z. But if they are incompatible, by properties of the interaction (Lemma 2.45) it means that $\Pi_1 p$ and $\Pi_1 p'$ are incompatible extensions of $\Pi_1 z$ mapping to the same event in the game, contradicting the $\sim$-receptivity of σ. Hence $p = p'$ and θ is the identity. □

As we will see later, even though there are non-trivial symmetries in the interaction of thin pre-$\sim$-strategies because the symmetries on $\mathcal{A}$ and $\mathcal{C}$ are not thin, whenever the interaction stays within B the phenomenon above applies, and the symmetry is trivial. In particular, a bijection in the symmetry of the interaction is fully determined by its restriction to visible events (cf. Lemma 3.28).

4.2. Composition of pre-$\sim$-strategies. We have seen how to adapt the interaction of pre-strategies in the presence of symmetry. We now move on to defining composition of pre-$\sim$-strategies. In this section we consider $\sigma : \mathcal{S} \nrightarrow \mathcal{A}^\perp \parallel \mathcal{B}$ and $\tau : \mathcal{T} \nrightarrow \mathcal{B}^\perp \parallel \mathcal{C}$ two pre-$\sim$-strategies.

4.2.1. *Interaction.* We apply Lemma 3.25 to the following pre-$\sim$-strategies:

$$\sigma \parallel \mathcal{C}^\perp : \mathcal{S} \parallel \mathcal{C}^\perp \nrightarrow \mathcal{A}^\perp \parallel \mathcal{B} \parallel \mathcal{C}^\perp \qquad \mathcal{A} \parallel \tau : \mathcal{A} \parallel \mathcal{T} \nrightarrow \mathcal{A} \parallel \mathcal{B}^\perp \parallel \mathcal{C},$$

resulting in the map

$$\tau \circledast \sigma = (\sigma \parallel \mathcal{C}^\perp) \wedge (\mathcal{A} \parallel \tau) : (\mathcal{S} \parallel \mathcal{C}) \wedge (\mathcal{A} \parallel \mathcal{T}) \nrightarrow \mathcal{A} \parallel \mathcal{B} \parallel \mathcal{C} \nrightarrow \mathcal{A} \parallel \mathcal{C}.$$

The resulting isomorphism family on $T \circledast S$ is thin:

LEMMA 3.27. *The isomorphism family on $T \circledast S$ is thin.*

PROOF. Let $z \in \mathscr{C}(T \circledast S)$ and $\theta : z \cup \{p\} \cong_{\widetilde{T \circledast S}} z \cup \{p'\}$ a nonnegative extension of id_z. At least one of $\Pi_1 p$ or $\Pi_2 p$ is nonnegative in S or T respectively. Assume for instance the former: $\Pi_1 p$ and $\Pi_1 p'$ correspond to events $s \in S$ and $s' \in S$ respectively. By definition of the symmetry on the pullback, θ induces $\theta_1 : \Pi_1(z \cup \{p\}) \cong_{\tilde{S} \parallel \tilde{C}} \Pi_1(z \cup \{p'\})$ which by projection on the S component yields an isomorphism $z_S \cup \{s\} \cong_{\tilde{S}} z_S \cup \{s'\}$ for some $z_S \in \mathscr{C}(S)$. Since $\tilde{S}$ is thin, $s = s'$ and $\Pi_1 p = \Pi_1 p'$.

If p and p' are compatible extensions of z, then by local injectivity of Π_1, $p = p'$ as desired. Otherwise it means that $\Pi_2 p$ and $\Pi_2 p'$ are incompatible extensions of $\Pi_1 z$. Since $\Pi_1(z \cup \{p, p'\})$ is consistent, this could only occur if $\Pi_2 p$ and $\Pi_2 p'$ lived in T (and were incompatible there). In that case, since $\Pi_1 p$ is nonnegative, it must be that it is actually positive and $\Pi_2 p$ negative in T. By $\sim$-receptivity of τ it must be that $\Pi_2 p = \Pi_2 p'$ as it should be unique extension of $\Pi_2 z$ by the event $(\tau \circledast \sigma)p$ of the game. □

4.2.2. *Hiding.* To deduce that essential events of $\tau \circledast \sigma$ are closed under symmetry, Via Lemma 3.18 the set $V = \{p \in T \circledast S \mid p \text{ is essential or visible}\}$ is closed under symmetry (as the union of two closed sets) so we can define $\mathcal{T} \circledcirc \mathcal{S} = \mathcal{T} \circledast \mathcal{S} \downarrow V$ and the obvious mapping $\tau \circledcirc \sigma : \mathcal{T} \circledcirc \mathcal{S} \to \mathcal{A}^\perp \parallel \mathcal{C}$.

It is however thin, as a consequence of the unique witness property: a symmetry between configurations of the composition induces a *unique* symmetry on their downclosure in the interaction:

LEMMA 3.28 (Unique witness). *Let $\sigma : \mathcal{S} \nrightarrow \mathcal{A}^\perp \parallel \mathcal{B}$ and $\tau : \mathcal{T} \nrightarrow \mathcal{B}^\perp \parallel \mathcal{C}$ be pre-$\sim$-strategies, and V the set of external events of $T \circledast S$ (ie. those sent to A or C).*

Let $\theta : x \cong_{\tilde{T} \circledast \tilde{S}} y$ and $\theta' : x \cong_{\tilde{T} \circledast \tilde{S}} y'$ such that $\theta \cap V^2 = \theta' \cap V^2$. Then $\theta = \theta'$.

PROOF. By hypothesis, we have that $y \cap V = y' \cap V$. Note that $\theta \circ \theta'^{-1} : y' \cong y \in (\tilde{S} \parallel \tilde{C}) \wedge (\tilde{A} \parallel \tilde{T})$ and contains $\mathrm{id}_{y \cap V}$. As a result, $\theta \circ \theta'^{-1}$ actually belongs to $(\tilde{S} \parallel C_=) \wedge (A_= \parallel \tilde{T})$ where $C_=$ and $A_=$ denote respectively the C and A adjoined with the trivial symmetry reduced to the identities bijections.

This is a pullback of thin pre-$\sim$-strategies, so $\theta \circ \theta'^{-1}$ is an identity bijection (Lemma 3.26) which implies $\theta = \theta'$. □

We can now deduce that $\tau \circledcirc \sigma$ is indeed a pre-$\sim$-strategy.

LEMMA 3.29. *Let $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}^\perp \parallel \mathcal{B}$ and $\tau : \mathcal{T} \rightharpoonup \mathcal{B}^\perp \parallel \mathcal{C}$ be pre-$\sim$-strategies. Then, $\tau \circledcirc \sigma$ is a pre-$\sim$-strategy.*

PROOF. Throughout the proof, we use $[\cdot]$ to abbreviate $[\cdot]_{T \circledast S}$.

Thin. Let $z \in \mathscr{C}(T \circledcirc S)$ such that id_z extends by (e, e') to $\theta : x \cong y \in \tilde{T} \circledcirc \tilde{S}$ with witness $[\theta] : [x] \cong_{T \circledast S} [y]$. Write θ_0 for $[\theta] \setminus \{(e, e')\} : x_0 \cong y_0$. By hypothesis, θ_0 behaves like the identity on the visible part of x_0. Hence, by Lemma 3.28, θ_0 is the identity on x_0. Since $\mathrm{id}_{x_0} = \theta_0$ can be extended to $\bar{\theta}$ by (e, e') which is nonnegative in $T \circledast S$, we can conclude by Lemma 3.27.

$\sim$-receptivity. Let $\theta : z \cong w$ in $\tilde{T} \circledcirc \tilde{S}$ with a negative extension p of z in $T \circledcirc S$. Assume moreover $(\tau \circledcirc \sigma)\theta$ extends by $((\tau \circledcirc \sigma)p, c)$ for $c \in A^\perp \parallel C$ that we consider in C without loss of generality.

Existence. In $T \circledast S$, we have the following extensions:

$$[z] \subseteq z' \overset{p}{\longrightarrow} \subset$$

where the first extension is only by inessential neutral events. Using extension of $\tilde{T} \circledast \tilde{S}$, $[\theta]$ extends to $\theta' : z' \cong_{\tilde{T} \circledast \tilde{S}} w'$ for some $w' \in \mathscr{C}(T \circledast S)$.

We now project on $\mathcal{A} \parallel \tau$: $(A \parallel \tau)(\Pi_2 \theta')$ extends by $(\Pi_2 p, c)$ which by $\sim$-receptivity of $\mathcal{A} \parallel \tau$, means that $\Pi_2 w'$ extends by $t' \in A \parallel T$ (actually it is in T) and $\Pi_2 \theta \cup \{(\Pi_2 p, t')\}$ is a valid extension of $\Pi_2 \theta$. It is easy to see that $\Pi_1 w'$ extends by $(2, c) \in S \parallel C$ as well. As a result, w' extends by an event $p' \in T \circledast S$ projecting to c in the game, and $\theta' \cup \{(p, p')\}$ is a valid symmetry of $\tilde{T} \circledast \tilde{S}$. As a result $\theta \cup \{(p, p')\}$ is a valid extension of θ in $\tilde{T} \circledcirc \tilde{S}$.

Uniqueness. Consider two such extensions of z, p_1 and p_2, and two extensions of θ, θ_1 and θ_2. The extensions $[z] \subseteq [z] \cup [p_1)$ and $[z] \subseteq [z] \cup [p_2)$ only contain inessential events, so they are compatible, and $z' = [z] \cup [p_1) \cup [p_2) \in \mathscr{C}(T \circledast S)$.

By the extension property, $[\theta]$ extends (only by inessential pairs) to $\theta' : w' \cong z'$ for some extension w' of $[w]$. Projecting to $\mathcal{A} \parallel \tau$, we get $\Pi_2 \theta' : \Pi_2 w' \cong \Pi_2 z'$. Since w' can still extend by p', $\Pi_2 w'$ can extend by $\Pi_2 p'$. Likewise, $\tau(\Pi_2 z')$ can be extended by c. However, z' can be extended in two ways: either by $\Pi_2 p_1$ or by $\Pi_2 p_2$, both of which give an extension of $\Pi_2 \theta$ defined on p. By $\sim$-receptivity of $\mathcal{A} \parallel \tau$, $\Pi_2 p_1 = \Pi_1 p_2$. Since $\Pi_1 p_1 = \Pi_1 p_2 = c$ as well, $p_1 = p_2$ as desired. □

Note that uniqueness for $\sim$-receptivity relies on essential events; indeed in the covered setting $\sim$-receptivity is not stable under composition on its own **[CCW14]**.

4.3. The category $\sim$-CG$^{\cong}_{\circledcirc}$. From there, we can construct a category, generalizing the constructions of the previous chapter. First, we define $\sim$-strategies as in Chapter 2:

DEFINITION 3.30. A $\sim$-strategy on a $\sim$-game $\mathcal{A}$ is a pre-$\sim$-strategy $\sigma : \mathcal{S} \to \mathcal{A}$ on $\mathcal{A}$ such that $\sigma : S \to A$ is an essential strategy.

To get a category, we still need to quotient by an equivalence relation. Isomorphism is generalized in a straightforward way:

DEFINITION 3.31 (Strong isomorphism). Let $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$ and $\sigma' : \mathcal{S}' \rightharpoonup A$ be pre-$\sim$-strategies. They are strongly isomorphic when there exists an isomorphism $\varphi : \mathcal{S} \cong \mathcal{S}'$ of event structures with symmetry making the usual triangle commute. In this case, we write $\sigma \cong \sigma'$.

This definition allows us to construct a category of uniform strategies, up to strong isomorphism:

PROPOSITION 3.32. *The following is a compact-closed category* $\sim -\mathrm{CG}^{\cong}_{\circledcirc}$*:*

- Objects: $\sim$*-games,*
- Morphisms form $\mathcal{A}$ to $\mathcal{B}$: $\sim$*-strategies on* $\mathcal{A}^{\perp} \parallel \mathcal{B}$ *up to strong isomorphism,*
- Composition: $\circledcirc$,
- Copycat: *the copycat enhanced with symmetry.*

PROOF. We omit the proof as this result is not needed in the rest of the document. However, Section 6 provides most of the arguments for the proof. □

5. Weak isomorphism

We have seen how to build a category of uniform strategies. However, strategies are still compared using strong isomorphism – in particular, the symmetry on the game plays no role in the equivalence relation on strategies. It is possible to change that by using the equivalence relation $\sim$ on maps of event structures with symmetry by asking that $\tau \circ \varphi \sim_{\mathcal{A}} \sigma$ instead of requiring an equality. However, proving that such an equality is a congruence is not easy. We cannot rely on the universal property of the interaction since our morphisms do not commute on the nose anymore. We need to generalize the universal property to account for symmetry.

Establishing the generalization proved very hard, and requires splitting $\mathcal{A}$ into a *negative* sub-isomorphism family $\mathcal{A}_-$ and a *positive* one $\mathcal{A}_+$, abstracting the setting for $!\widetilde{A}$ that can be split into a negative and a positive part. This is reminiscent of Melliès' notion of uniformity [**Mel03**] in terms of bi-invariance under group actions, the two groups being Opponent reindexings and Player reindexings. Axiomatizing this decomposition leads to the notion of *thin concurrent game*:

DEFINITION 3.33. A **thin concurrent game (tcg)** is an essp $\mathcal{A}$ with two additional isomorphism families $\tilde{A}_-$ and $\tilde{A}_+$ on A such that:

(a) The families $\tilde{A}_+$ and $\tilde{A}_-$ are subsets of $\tilde{A}$,
(b) If $\theta \in \tilde{A}_+ \cap \tilde{A}_-$ then θ is an identity bijection,
(c) If $\theta \in \tilde{A}_-$ and $\theta \subseteq^- \theta' \in \tilde{A}$ then $\theta' \in \tilde{A}_-$,
(d) If $\theta \in \tilde{A}_+$ and $\theta \subseteq^+ \theta' \in \tilde{A}$ then $\theta' \in \tilde{A}_+$.

where $\theta \subseteq^- \theta'$ (resp. $\theta \subseteq^+ \theta'$) means that $\theta \subseteq \theta'$ such that $\theta' \setminus \theta$ only contains events of negative (resp. positive) polarity. The triple $(\mathcal{A}, \tilde{A}_-, \tilde{A}_+)$ will be often written simply $\mathcal{A}$ to ease notation.

In general, from a $\sim$-game, one cannot extract such a decomposition because of pathological cases. However, if one restricts to *local* isomorphism families – those generated by an equivalence relation on events – one can recover the decomposition. Remark that this definition does not ask that tcgs be $\sim$-games: indeed, it follows from the axioms (Lemma 3.37).

Given a tcg $\mathcal{A}$, we will write $\mathcal{A}_-$ for the event structure with symmetry $(A, \tilde{A}_-)$ and $\mathcal{A}_+$ for $(A, \tilde{A}_+)$. Thin concurrent games support the usual operation on games:

DEFINITION 3.34. Given a tcg $(\mathcal{A}, \tilde{A}_-, \tilde{A}_+)$, its **dual** is

$$(\mathcal{A}, \tilde{A}_-, \tilde{A}_+)^\perp = (\mathcal{A}^\perp, \tilde{A}_+, \tilde{A}_-)$$

Note that the two additional isomorphism families are swapped.

Likewise, the **simple parallel composition** of $(\mathcal{A}, \tilde{A}_-, \tilde{A}_+)$ and $(\mathcal{B}, \tilde{B}_-, \tilde{B}_+)$ is performed component-wise:

$$(\mathcal{A}, \tilde{A}_-, \tilde{A}_+) \parallel (\mathcal{B}, \tilde{B}_-, \tilde{B}_+) = (\mathcal{A} \parallel \mathcal{B}, \tilde{A}_- \parallel \tilde{B}_-, \tilde{A}_+ \parallel \tilde{A}_+)$$

where parallel composition of sets of bijections is defined as in Definition 3.7.

5.1. The decomposition lemma. We first make formal the intuition of decomposition by showing that symmetries in $\mathcal{A}$ uniquely decompose into a part in $\mathcal{A}_-$ and a part in $\mathcal{A}_+$.

LEMMA 3.35. *Let $\mathcal{A}$ be a tcg and $x \in \mathscr{C}(A)$.*

- *If $\mathrm{id}_x \subseteq^+ \theta \in \tilde{A}_-$ then $\theta = \mathrm{id}_y$ for some $x \subseteq y \in \mathscr{C}(A)$.*
- *If $\mathrm{id}_x \subseteq^- \theta \in \tilde{A}_+$ then $\theta = \mathrm{id}_y$ for some $x \subseteq y \in \mathscr{C}(A)$.*

PROOF. The two items are dual; we only detail the first. Since $\mathrm{id}_x \in \tilde{A}_+$, (d) entails that $\theta \in \tilde{A}_+$, so $\theta \in \tilde{A}_+ \cap \tilde{A}_-$, hence the conclusion follows by (b). □

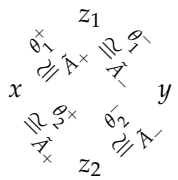
PROPOSITION 3.36 (Decomposition lemma). *Let $\mathcal{A}$ be a tcg. The following function is an order-isomorphism:*

$$\begin{array}{rcl} \tilde{A}_- \times_A \tilde{A}_+ & \to & \tilde{A} \\ (\theta^-, \theta^+) & \mapsto & \theta^- \circ \theta^+ \end{array}$$

where $\tilde{A}_- \times_A \tilde{A}_+ = \{(\theta^-, \theta^+) \in \tilde{A}_- \times \tilde{A}_+ \mid \mathrm{codom}\,\theta^+ = \mathrm{dom}\,\theta^-\}$ is ordered by pairwise inclusion and $\tilde{A}$ is ordered by inclusion.

PROOF. The map is well defined because $\tilde{A}_-$ and $\tilde{A}_+$ are included in $\tilde{A}$.

Injectivity. Assume we have $\theta = \theta_1^- \circ \theta_1^+ = \theta_2^- \circ \theta_2^+ : x \cong_{\tilde{A}} y$. In other words we have the following commutative square:



By using groupoid laws we get that $\theta_1^+ \circ (\theta_2^+)^{-1} = (\theta_1^-)^{-1} \circ \theta_2^- : z_1 \cong z_2 \in \tilde{A}_- \cap \tilde{A}_+$ hence it is equal to the identity: $z_1 = z_2$, $\theta_1^+ = \theta_2^+$ and $\theta_1^- = \theta_2^-$.

Surjectivity. By induction on $\theta \in \tilde{A}$ we build a pre-image. If θ is empty then $(\emptyset, \emptyset)$ is suitable.

Assume we have the decomposition of $\theta : x \cong_{\tilde{A}} y$ into $x \overset{\theta^+}{\cong}_{\tilde{A}_+} z \overset{\theta^-}{\cong}_{\tilde{A}_-} y$ and θ extends to $\theta' : x' \cong y'$ by a pair of fixed polarity, say positive. We use the extension axiom on θ^- to get $\theta^- \subseteq \theta'^- : z' \cong_{\tilde{A}_-} y'$. It follows that $\theta^+ \subseteq (\theta'^-)^{-1} \circ \theta' : x' \cong_{\tilde{A}} z'$

is a positive extension of θ^+ so it must belong to $\tilde{A}_+$ by the properties of tcgs. Hence $\theta' = \theta'^- \circ \left((\theta'^-)^{-1} \circ \theta'\right)$ provides the required decomposition.

Monotonicity and monotonicity of the inverse. Clearly, the function is monotonic. We prove that so is its inverse. Assume $\theta_- \circ \theta_+ \subseteq \theta'_- \circ \theta'_+$. We write $\theta_- \circ \theta_+ : x \cong_{\tilde{A}} y$ and $\theta'_- \circ \theta'_+ : x' \cong_{\tilde{A}} y'$; in particular we have $x \subseteq x'$ and $y \subseteq y'$. But then restricting θ'_+ to x yields $\theta''_+ : x \cong_{\tilde{A}_+} z$, and restricting θ'_- to z yields $\theta''_- : z \cong_{\tilde{A}_-} y$, where $\theta''_- \circ \theta''_+$ is the restriction of $\theta'_- \circ \theta'_+$ to x, *i.e.* $\theta_- \circ \theta_+$. By injectivity (proved above), $\theta_- = \theta''_-$ and $\theta_+ = \theta''_+$. Thus, $\theta_- \subseteq \theta'_-$ and $\theta_+ \subseteq \theta'_+$. □

As a consequence of this lemma, the following commutative diagram:

$$\begin{array}{ccccc} & & (A, \{\mathrm{id}_x \mid x \in \mathscr{C}(A)\}) & & \\ & \swarrow & & \searrow & \\ (A, \tilde{A}_+) & & & & (A, \tilde{A}_-) \\ & \searrow & & \swarrow & \\ & & \mathcal{A} & & \end{array}$$

is a both a pullback and a push-out in $\mathcal{E}^\sim$. In particular, this means that $\tilde{A}$ is uniquely determined from $\tilde{A}_-$ and $\tilde{A}_+$.

This decomposition is particularly useful to establish that thin concurrent games are special cases of $\sim$-games:

LEMMA 3.37. *Let $\mathcal{A}$ be a tcg. Then $\mathcal{A}$ is a $\sim$-game.*

PROOF. We first prove that $\tilde{A}_+$ and $\tilde{A}_-$ are $\sim$-games. Let $\theta : x \cong_{\tilde{A}_+} y$ with a positive extension $\theta_1 : x_1 \cong_{\tilde{A}_+} y_1$ and a negative extension $\theta_2 : x_2 \cong_{\tilde{A}_+} y_2$. Lemma 2.59 implies $x_1 \cup x_2 \in \mathscr{C}(A)$.

Using (Extension) of $\tilde{A}_+$ twice to θ_1 and θ_2, we get to the following picture:

$$\begin{array}{ccc} \theta'_1 : x_1 \cup x_2 \cong_{\tilde{A}_+} y'_1 & & \theta'_2 : x_1 \cup x_2 \cong_{\tilde{A}_+} y'_2 \\ \mathrel{\rotatebox{90}{$\subseteq$}} \quad \cup\mathrel{|} & & \mathrel{|}\cup \quad \mathrel{\rotatebox{90}{$\subseteq$}} \\ \theta_1 : x_1 \cong_{\tilde{A}_+} y_1 & & \theta_2 : x_2 \cong_{\tilde{A}_+} y_2 \\ & \times \swarrow \quad \searrow & \\ & \theta : x \cong_{\tilde{A}_+} y & \end{array}$$

By the (Groupoid) axiom on $\tilde{A}_+$, we have $\mathrm{id}_y \subseteq \theta'_1 \circ {\theta'_2}^{-1} : y'_2 \cong_{\tilde{A}_+} y'_1$. By (Restriction), we build $\varphi = \theta'_1 \circ {\theta'_2}^{-1} \restriction y_2$. By construction, we have $\mathrm{id}_y \subseteq^- \varphi \in \tilde{A}_+$, so $\varphi = \mathrm{id}_{y_2}$ by Lemma 3.35. It follows that $\theta_2 \subseteq \theta'_1$, hence $\theta'_1 = \theta_1 \cup \theta_2$ as required. A dual reasoning shows that $\tilde{A}_-$ is race-preserving as well.

Now, we deduce the result for $\tilde{A}$, using the decomposition of Lemma 3.36. Assume $\theta = \theta^- \circ \theta^+$ has extensions $\theta \subseteq^+ \theta_1$ and $\theta \subseteq^- \theta_2$, with decompositions $\theta_1^- \circ \theta_1^+$ and $\theta_2^- \circ \theta_2^+$. By monotonicity of the decomposition, we have $\theta^+ \subseteq^+ \theta_1^+$, $\theta^+ \subseteq^- \theta_2^+$, $\theta^- \subseteq^+ \theta_1^-$ and $\theta^- \subseteq^- \theta_2^-$. By race-freeness of $\tilde{A}_+$ it follows first that $\theta_1^+ \cup \theta_2^+ \in \tilde{A}_+$, and then by race-freeness of $\tilde{A}_-$ it follows that $\theta_1^- \cup \theta_2^- \in \tilde{A}_-$. Thus

$$(\theta_1^- \cup \theta_2^-) \circ (\theta_1^+ \cup \theta_2^+) = (\theta_1^- \circ \theta_1^+) \cup (\theta_2^- \circ \theta_1^-) = \theta_1 \cup \theta_2 \in \tilde{A}. \quad \square$$

5.2. Weak isomorphism and the bipullback property. A (pre-)$\sim$-strategy on a tcg $\mathcal{A}$ is a (pre-)$\sim$-strategy on the underlying $\sim$-game $\mathcal{A}$. We keep the notation $\sigma : \mathcal{A}, \sigma : \mathcal{S} \rightarrowtail \mathcal{A}$ when $\mathcal{A}$ is a tcg to introduce (pre-)$\sim$-strategies.

We can now define a coarser equivalence relation, weak isomorphism, that compares strategies up to symmetry induced by the game.

DEFINITION 3.38 (Weak isomorphism). Two pre-$\sim$-strategies $\sigma : \mathcal{S} \rightarrowtail \mathcal{A}$ and $\tau : \mathcal{T} \rightarrowtail \mathcal{A}$ are **weakly isomorphic** (written $\sigma \cong \tau$) when there exists an isomorphism of event structure with symmetry $\varphi : \mathcal{S} \cong \mathcal{T}$ such that $\tau \circ \varphi \sim_{\mathcal{A}_+} \sigma$.

Remark that the definition uses the positive symmetry – $\mathcal{A}_+$ – instead of the full symmetry. Both definitions are equivalent when instantiated to $!A$, but this definition turns out to have fruitful consequences (eg. Lemma 3.40).

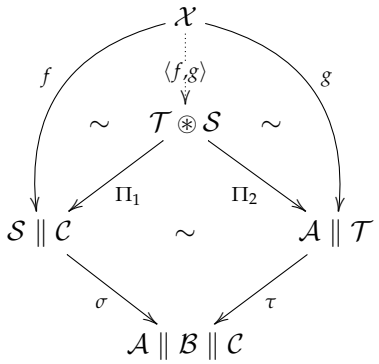
The corresponding triangle is said to commute *up to symmetry*. We use the abbreviation $\sim^+$ for $\sim_{\mathcal{A}_+}$ when the corresponding $\mathcal{A}$ is clear from context.

To prove that $\cong$ is a congruence, we need to show that $\mathcal{T} \circledast \mathcal{S}$ satisfies a universal property up to symmetry, which we call the *bipullback property* – as in the covered case this universal property coincides with that of a bipullback.

PROPOSITION 3.39 (Bipullback property). *Let $\sigma : \mathcal{S} \rightarrowtail \mathcal{A}^\perp \parr \mathcal{B}$ and $\tau : \mathcal{T} \rightarrowtail \mathcal{B}^\perp \parr \mathcal{C}$ be pre-$\sim$-strategies. The ess $\mathcal{T} \circledast \mathcal{S}$ enjoys the following universal property: for all $f : \mathcal{X} \rightarrowtail \mathcal{S} \parr \mathcal{C}$ and $g : \mathcal{X} \rightarrowtail \mathcal{A} \parr \mathcal{T}$ not synchronizing on neutral events, such that $\tau \circ g \sim_{\tilde{A} \parr \tilde{B} \parr \tilde{C}} \sigma \circ f$, there exists $\langle f, g \rangle : \mathcal{X} \rightarrowtail \mathcal{T} \circledast \mathcal{S}$, unique up to symmetry, with*

$$\Pi_1 \circ \langle f, g \rangle \sim_{\tilde{S} \parr \tilde{C}_+} f \qquad \text{and} \qquad \Pi_2 \circ \langle f, g \rangle \sim_{\tilde{A}_- \parr \tilde{T}} g.$$

This is summed up by the following diagram (where all squares and triangle commutes up to $\sim$ in the category of event structures with symmetry):



The proof of this result is delayed until the next subsection. We finish this subsection by showing how this result implies that weak isomorphism is a congruence. To do that, we notice that to build a weak isomorphism, it is enough to build a *weak equivalence*, that is a pair of maps that are inverse of each other only up to symmetry:

LEMMA 3.40 (Weak equivalence lemma). *Let $\sigma : \mathcal{S} \rightarrowtail \mathcal{A}$ and $\sigma' : \mathcal{S}' \rightarrowtail \mathcal{A}$ be pre-$\sim$-strategies. Assume there exists $f : \mathcal{S} \to \mathcal{T}$ and $g : \mathcal{T} \to \mathcal{S}$ with $\sigma' \circ f \sim_{\mathcal{A}_+} \sigma$ and $g \circ f \sim_{\mathcal{S}} \mathrm{id}_{\mathcal{S}}$ as well as $f \circ g \sim_{\mathcal{T}} \mathrm{id}_{\mathcal{T}}$. Then $f \circ g = \mathrm{id}_T$ and $g \circ f = \mathrm{id}_S$ and as a consequence (f, g) is a weak isomorphism between σ and σ'.*

This lemma is necessary because the bipullback property (via its uniqueness up to symmetry) naturally permits constructing weak equivalences rather than weak isomorphisms. The proof of this lemma relies heavily on the thin hypothesis.

PROOF. By hypothesis, for all $x \in \mathscr{C}(S)$, the canonical bijection $\theta_x : x \cong g\,(f\,x)$ is in $\tilde{S}$. We show by induction on x it is always the identity hence $g \circ f = \mathrm{id}_{\mathcal{S}}$.

The base case is trivial. Assume the result for $x \in \mathscr{C}(S)$ and suppose x extends by $s \in S$ to x'. If s is nonnegative, then $\theta_x = \mathrm{id}_x$ extends by $(s, \theta_{x'}\, s)$ which is also nonnegative, so since $\tilde{S}$ is thin we have $s = \theta_{x'}\, s$ as desired.

If s is negative, we know that $\tau \circ f \sim^+ \sigma$ and as a result $\tau \sim^+ \sigma \circ g$. Hence $\sigma \sim^+ \sigma \circ g \circ f$. As a result, the obvious $\varphi_{x'} : \sigma_{\downarrow}\, x' \cong \sigma_{\downarrow}\,(g\,(f\,x'))$ is in $\tilde{A}_+$. By induction hypothesis, we know it is an extension of the identity on $\sigma_{\downarrow} x$. Hence $\mathrm{id}_{\sigma_{\downarrow} x}$ extends in $\tilde{A}_+$ with $(\sigma\, s, \sigma\,(g\,(f\,s)))$ with s negative, so $\sigma s = \sigma\,(g\,(f\,s))$ by Lemma 3.35. By $\sim$-receptivity of σ it follows that $s = g\,(f\,s) = \theta_{x'}\, s$. □

This lemma is crucial to deduce that maps obtained from the universal property of Proposition 3.39 preserves essential events:

LEMMA 3.41. *For $f : \mathcal{A} \to \mathcal{B}$ and $g : \mathcal{B} \to \mathcal{A}$ maps of* thin essps*, such that*

$$g \circ f \sim_{\mathcal{A}} \mathrm{id}_{\mathcal{A}} \qquad \textit{and} \qquad f \circ g \sim_{\mathcal{B}} \mathrm{id}_{\mathcal{B}},$$

then f and g preserve nonnegative incompatible extensions.

PROOF. Let $x \in \mathscr{C}(A)$ and a_1, a_2 be two nonnegative incompatible extensions of x. Assume that fa_1 and fa_2 are compatible extensions of fx. Then since $gf(x \cup \{a_1\})$ can extend by gfa_2 and $x \cup \{a_1\} \cong gf(x \cup \{a_1\})$ given by $g \circ f \sim_{\mathcal{A}} \mathrm{id}_{\mathcal{A}}$, $x \cup \{a_1\}$ extends by some a_2'. This implies that $\mathrm{id}_x \in \tilde{A}$ extends by (a_2, a_2') (by composition of restrictions of the previous isomorphisms). Since a_2 and a_2' are nonnegative, it follows that $a_2 = a_2'$ by thin. This contradicts the fact that $x \cup \{a_1, a_2\}$ is not consistent. □

All the ingredients are there to prove that weak isomorphism is a congruence:

PROPOSITION 3.42. *Weak isomorphism is a congruence.*

PROOF. Let $\sigma, \sigma : \mathcal{S} \nrightarrow A^{\perp} \parallel \mathcal{B}$ be pre-$\sim$-strategies and $\varphi : \sigma \cong \sigma'$. Let $\tau : \mathcal{T} \nrightarrow \mathcal{B}^{\perp} \parallel \mathcal{C}$ and $\tau' : \mathcal{T}' \nrightarrow \mathcal{B}^{\perp} \parallel \mathcal{C}$ be other pre-$\sim$-strategies and $\psi : \tau \cong \tau'$.

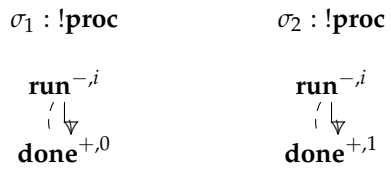
We first build a map $\mu : \mathcal{T} \circledast \mathcal{S} \to \mathcal{T}' \circledast \mathcal{S}'$ as:

$$\mathcal{T} \circledast \mathcal{S} \xrightarrow{\langle (\mathcal{A} \| \psi) \circ \Pi_2, (\varphi \| \mathcal{C}) \circ \Pi_1 \rangle} \mathcal{T}' \circledast \mathcal{S}'$$

(well-defined by Proposition 3.39). Conversely there is a map $\nu : T' \circledast S' \to T \circledast S$. By the uniqueness up to symmetry, we get that $\mu \circ \nu \sim_{\mathcal{T}' \circledast \mathcal{S}'} \mathrm{id}_{T' \circledast S'}$ and $\nu \circ \varphi \sim_{\mathcal{T} \circledast \mathcal{S}} \mathrm{id}_{\mathcal{T} \circledast \mathcal{S}}$. By Lemma 3.41, it follows that μ and ν preserve essential events. They induce maps $\bar{\mu} : T \circledcirc S \to T \circledcirc S'$ and $\bar{\nu} : T \circledcirc S' \to T \circledcirc S$ that are still inverse of each other up to symmetry and commute on the game up to symmetry. Thus they meet the conditions of Lemma 3.40 and actually form a weak isomorphism $\tau \circledcirc \sigma \approxeq \tau \circledcirc \sigma'$. □

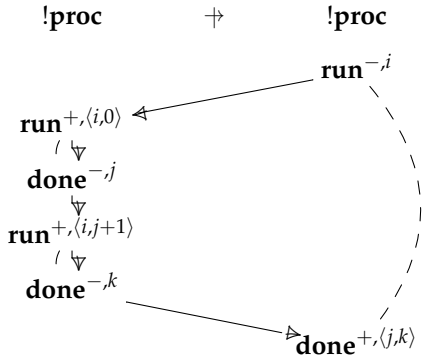
5.3. Proof of the bipullback property. We end this section by a detailed proof of Proposition 3.39. The proof relies on the observation of Lemma 3.26: interactions of dual pre-$\sim$-strategies have trivial symmetries. The difficulty lies in the existence part of the universal property, as the following example demonstrates:

EXAMPLE 3.43. Consider the two following strategies on !**proc**:



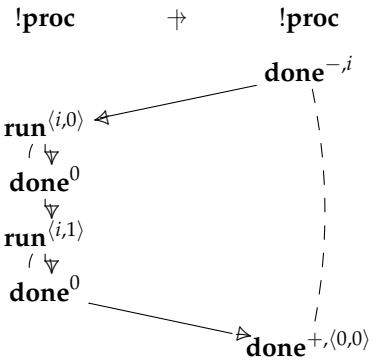
that only differ by the choice of copy index for **done**. For each of them, there is a unique symmetry that make them strong-receptive.

There is a weak isomorphism $\varphi : \sigma_1 \cong \sigma_2$. Consider the following strategy τ:



which represents x : **proc** $\vdash$ $x; x$: **proc**. The $\langle\cdot\rangle$ symbol stands for any injection $\mathbb{N}^* \to \mathbb{N}$ needed to avoid index collision and thus guarantee local injectivity (more on that in the next chapter, see Example 4.4).

In order to build a weak isomorphism between the resulting compositions $\tau \circledcirc \sigma_1$ and $\tau \circledcirc \sigma_2$, a reasonable first step is to build a weak isomorphism between the interactions $\tau \circledast \sigma_1$ and $\tau \circledast \sigma_2$. In particular, given a configuration of $T \circledast S_1$, we should be able to build a canonical configuration of $T \circledast S_2$. Consider *e.g.* the following configuration of $T \circledast S_1$.



where events on the left hand side are drawn without polarity, as they are synchronized between σ_1 and τ. It is easy to extract from this representation configurations $x \in \mathscr{C}(S_1 \parallel !\mathbf{proc})$ and $y \in \mathscr{C}(T)$ such that

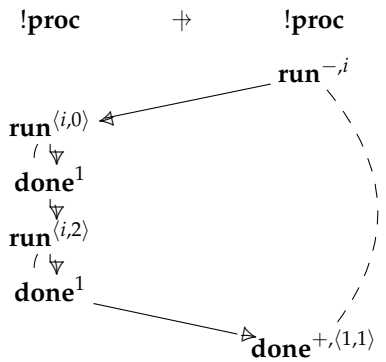
$$(\sigma_1 \parallel !\mathbf{proc})\, x = \tau\, y$$

and such that the induced bijection is secured.

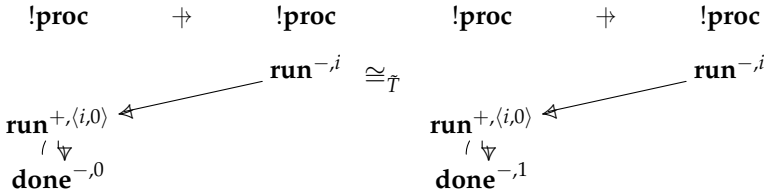
In order to construct a configuration in $T \circledast S_2$, it is natural to try and replace x with $\varphi(x)$ – and that would work out if φ was a *strong* isomorphism. But as it is only a weak isomorphism, we do not have $(\sigma_2 \parallel !\mathbf{proc})\,(\varphi\, x) = \tau\, y$, only

$$(\sigma_2 \parallel !\mathbf{proc})\,(\varphi\, x) \cong_{\widetilde{!\mathbf{proc} \parallel !\mathbf{proc}}} \tau\, y$$

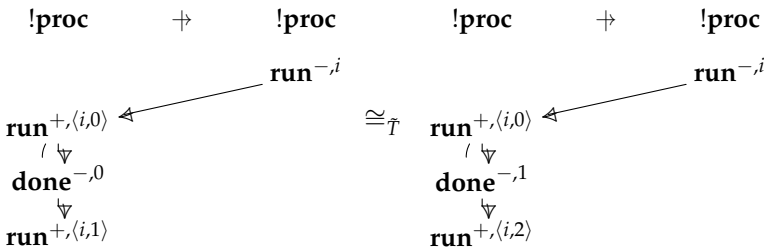
However, we can indeed extract from $\varphi\, x$ and y a valid configuration of $T \circledast S_2$. For our example, the only possibility is:



It appears that *both* $\varphi\, x$ and y had to change, in order to find an agreement as to the choice of copy indices. Firstly, by $\sim$-receptivity, $\tilde{T}$ comprises a bijection:



By (Extension) in $\tilde{T}$, we know that this bijection can be extended to some:



Likewise, by $\sim$-receptivity of $\sigma_2 \parallel !\mathbf{proc}$ this extension is lifted to $\tilde{S_2} \parallel \widetilde{!\mathbf{proc}}$, and we then apply (Extension) on $\tilde{S_2}$. And the process goes on, interactively between σ_2 and τ, until we get $x' \cong_{\tilde{S} \parallel \widetilde{!\mathbf{proc}}} \varphi\, x$ and $y' \cong_{\tilde{T}} y$ such that $(\sigma_2 \parallel !\mathbf{proc})\, x' = \tau\, y'$ (which in our example, is the configuration of the interaction above).

Formalizing this process of using $\sim$-receptivity on one strategy and extension on the other yields the following lemma:

LEMMA 3.44 (Weak bipullback property). *Let $\sigma : \mathcal{S} \nrightarrow \mathcal{A}$ and $\tau : \mathcal{T} \nrightarrow \mathcal{A}^\perp$ be $\sim$-receptive partial maps of event structures with symmetry. Let $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$ and $\theta : \sigma x \cong_{\tilde{A}} \tau y$, such that the composite bijection*

$$x \parallel y_\star \simeq \sigma\, x \parallel x_\star \parallel y_\star \overset{\theta}{\cong} x_\star \parallel \tau\, y \parallel y_\star \simeq x_\star \parallel y$$

is secured. Then, there exists $z \in \mathscr{C}(S \wedge T)$ along with $\theta_S : x \cong_{\tilde{S}} \Pi_1 z$ and $\theta_T : \Pi_2 z \cong_{\tilde{T}} y$, such that $\tau\theta_T \circ \sigma\theta_S = \theta$. Moreover, z is unique up to symmetry.

PROOF. *Uniqueness.* Assume we have such (z, θ_S, θ_T) and $(z', \theta'_S, \theta'_T)$. Then it is easy to see that $\theta'_S \circ \theta_S^{-1} : \Pi_1 z \cong_{\tilde{S}} \Pi_1 z'$ and similarly $\theta'_T \circ \theta_T^{-1} : \Pi_2 z \cong_{\tilde{T}} \Pi_2 z'$. Those match on the game $\mathcal{A}$, so they induce a $z \cong z'$ in $\tilde{S} \wedge \tilde{T}$ as desired.

Existence. We proceed by induction on θ; the base case is trivial. Assume θ extends by $(\sigma s, \tau t)$ to $\theta' : \sigma x' \cong \tau y'$. For instance, s is positive. By induction, we have $\theta_S : x \cong \Pi_1 z$, and x can be extended to $x' := x \cup [s]$, so by the extension property of the symmetry θ_S extends to $\theta'_S : x' \cong z'_S$. This means that $\tau\,\theta_T$ can be extended by symmetric negative (for T) events so by $\sim$-receptivity, θ_T can extend to $\theta'_T : z'_T \cong_{\tilde{T}} y'$, with $\sigma z'_S = \tau z'_T$ by construction. Since the bijection $z'_S \cong z'_T$ is obviously secured, we get $z' \in \mathscr{C}(S \wedge T)$ that satisfies our property. □

To build a map, uniqueness up to symmetry is not enough, however since our symmetries on pre-$\sim$-strategies are thin, uniqueness on the nose follows from Lemma 3.26. Mapification (Lemma 3.44) gives us a way to define a map on configurations instead of directly on the events. We can now assemble the pieces together and prove the bipullback property:

PROOF. (Of Proposition 3.39).

Uniqueness. Assume there are two such maps $\omega, \omega' : \mathcal{X} \to \mathcal{T} \circledast \mathcal{S}$. Let $x \in \mathscr{C}(X)$. The induced bijection $\Pi_1\,(\omega\, x) \cong \Pi_1\,(\omega'\, x)$ is in $\tilde{S} \parallel \tilde{C}$ as the composition

$$\Pi_1\,(\omega\, x) \cong_{\tilde{S}\parallel\tilde{C}} f\, x \cong_{\tilde{S}\parallel\tilde{C}} \Pi_1\,(\omega'\, x)$$

Similarly $\Pi_2\,(\omega\, x) \cong \Pi_2\,(\omega'\, x) \in \tilde{A} \parallel \tilde{T}$ and those two bijections match on the game, hence $\omega\, x \cong \omega'\, x \in \tilde{T} \circledast \tilde{S}$ which means $\omega \sim_{\tilde{T}\circledast\tilde{S}} \omega'$.

Existence. Consider the following pre-$\sim$-strategies:

$$(\sigma \parallel \mathcal{C}_+) : \mathcal{S} \parallel \mathcal{C}_+ \to \mathcal{A}^\perp \parallel \mathcal{B} \parallel \mathcal{C}^\perp \qquad (\mathcal{A}_- \parallel \tau) : \mathcal{A}_- \parallel \mathcal{T} \to \mathcal{A} \parallel \mathcal{B}^\perp \parallel \mathcal{C}$$

Those are $\sim$-strategies hence their pullback has a trivial symmetry by Lemma 3.26. This pullback has the same events as $\tilde{T} \circledast \tilde{S}$, only fewer symmetries.

Let $x \in \mathscr{C}(X)$. By the above remark and Lemma 3.44 applied to $f\, x \in \mathscr{C}(S \parallel C)$ and $g\, x \in \mathscr{C}(A \parallel T)$ we get a unique $z \in \mathscr{C}(T \circledast S)$ with $\Pi_1 z \cong_{\tilde{S}\parallel\tilde{C}_+} f\, x$ and $\Pi_2 z \cong_{\tilde{A}_-\parallel\tilde{T}} g\, x$. This construction induces a map $\psi : \mathscr{C}(X) \to \mathscr{C}(T \circledast S)$ such that $\Pi_1\,(\psi\, x) \cong f\, x$ and $\Pi_2\,(\psi\, x) \cong g\, x$.

To conclude we prove that ψ satisfies the conditions from Lemma 2.48, which follows from ψ being monotonic and preserving cardinality. Hence, we define $\langle f, g \rangle$ to the map of event structures induced by ψ. It preserves symmetry and satisfies the desired equivalence by construction. □

6. A compact-closed category

Since weak isomorphism is a congruence, we can build a category of $\sim$-strategies up to weak isomorphism: This definition leads to a

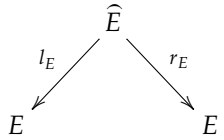
THEOREM 3.45. *The following data forms a compact-closed category* $\sim$-$\mathrm{tCG}^{\cong}_{\circledcirc}$*:*

- Objects*: thin concurrent games,*
- Morphisms from $\mathcal{A}$ to $\mathcal{B}$*:* $\sim$*-strategies from* $\mathcal{A}$ *to* $\mathcal{B}$ *up to weak isomorphism,*
- Composition*: The composition operator* $\circledcirc$ *on pre-*$\sim$*-strategies,*
- Copycat*: The copycat pre-*$\sim$*-strategy* $\mathfrak{c}_A$*.*

To prove this result, we need to establish the categorical laws (identity and associativity) as well as the compact-closed structure. However, structurally, this category is very close to $\mathrm{CG}_{\circledcirc}$. Indeed, there is a way to regard an isomorphism family $\tilde{A}$ on A as an event structure $\widehat{A}$ along with maps $l_A, r_A : \widehat{A} \to A$ satisfying various properties. Such structures will be called **spans**. It turns out that the structure of games and strategies neatly decompose at those two levels (A and $\widehat{A}$) and allows us to import the results from Chapter 2.

In this section, we follow the proofs of **[CCW14]**.

6.1. From event structures with symmetry to spans of event structures. Event structures with symmetry were first introduced as spans of event structures



satisfying some further properties: l_E, r_E are jointly monic, they are *open maps* **[Win07]**, and they satisfy the diagrams of (categorical) equivalence relations. The detail of these conditions will not be useful here; however we will use that event structures with symmetry can be represented as spans of event structures:

DEFINITION 3.46 (Span of event structures). A **span of event structures** is a tuple $\mathbb{A} = (\widehat{A}, A, l_A : \widehat{A} \to A, r_A : \widehat{A} \to A)$.

We also write spans as: $\mathbb{A} = A \xleftarrow{l_A} \widehat{A} \xrightarrow{r_A} A$.

We review how to go from an isomorphism family to a span. Let $\mathcal{E} = (E, \tilde{E})$ be an event structure with symmetry. To turn $\tilde{E}$ into an event structure we perform a construction similar to the *prime construction* used to turn the space of interaction states into an event structure. Define the event structure $\Pr(\tilde{E})$ as follows:

- *Events*: those bijections in $\tilde{E}$ between prime configurations $\theta : [e] \cong [e']$ (called *prime symmetries*),
- *Causality*: inclusion of bijections,
- *Consistency*: a finite set of prime symmetries Θ is consistent when there exists $\theta' \in \tilde{E}$ such that for all $\theta \in \Theta$, $\theta \subseteq \theta'$.

This event structure comes with two maps $l_E, r_E : \Pr(\tilde{E}) \to E$ mapping $\theta : [e] \cong [e']$ to e and e' respectively. The axioms of event structures are direct to check, along with the fact that l_E, r_E are maps of event structures. Note also that this definition extends in the presence of polarities in a straightforward way.

Just like for interactions (Lemma 2.44, configurations of $\Pr(\tilde{E})$ are in correspondence with symmetries via a canonical isomorphism $\mathscr{C}(\Pr(\tilde{E})) \cong \tilde{E}$.

A **partial map** of spans $f : \mathbb{A} \rightharpoonup \mathbb{B}$ is a pair $(f : A \rightharpoonup B, \widehat{f} : \widehat{A} \rightharpoonup \widehat{B})$ such that

$$\begin{array}{ccccc} A & \xleftarrow{l_A} & \widehat{A} & \xrightarrow{r_A} & A \\ \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle \widehat{f}} & & \downarrow{\scriptstyle f} \\ B & \xleftarrow[l_B]{} & \widehat{B} & \xrightarrow[r_B]{} & B \end{array}$$

6.2. Spanning games and strategies. By moving from event structures to spans of event structures, we can reproduce the construction of Chapter 2. A span of event structures with partial (resp. total) polarities is a span $(A, \widehat{A}, l_A, r_A)$ where A and $\widehat{A}$ carry partial (resp. total) polarities that are preserved by l_A and r_A.

A **span-game** is a span $\mathbb{A} = A \xleftarrow{l_A} \widehat{A} \xrightarrow{r_A} A$ with total polarities such that A and $\widehat{A}$ are race-free. The constructions of dual and parallel composition naturally component-wise extend to span-games.

Call a span with partial polarities **thin** when the projection maps preserve nonnegative incompatible extensions. A **span-strategy** on a span-game $\mathbb{A}$ is simply a partial map $\mathbb{S} \rightharpoonup \mathbb{A}$ where $\mathbb{S}$ is a *thin* span of event structures (polarities are induced by the mapping). The thin condition plays a similar role as for $\sim$-strategies, to ensure in particular that essential events are closed under symmetry in the interaction.

Likewise, a span-strategy from a span-game $\mathbb{A}$ to a span-game $\mathbb{B}$ is simply a partial map $(\sigma, \widehat{\sigma}) : \mathbb{S} \rightharpoonup \mathbb{A}^\perp \parallel \mathbb{B}$ such that both $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\widehat{\sigma} : \widehat{S} \rightharpoonup \widehat{A}^\perp \parallel \widehat{B}$ are essential strategies and $\mathbb{S}$ is thin.

6.2.1. *Copycat.* The copycat construction $A \mapsto \mathbb{CC}_A$ extends to a functor $\mathcal{E} \to \mathcal{E}$, mapping $f : A \to B$ to $CC_f : \mathbb{CC}_A \to \mathbb{CC}_B$, acting as $f \parallel f$ on events. With this remark, the following span can be checked to be a span-strategy:

$$\begin{array}{ccccc} \mathbb{CC}_A & \xleftarrow{\mathbb{CC}_{l_A}} & \mathbb{CC}_{\widehat{A}} & \xrightarrow{\mathbb{CC}_{r_A}} & \mathbb{CC}_A \\ \downarrow{\scriptstyle \alpha_A} & & \downarrow{\scriptstyle \alpha_{\widehat{A}}} & & \downarrow{\scriptstyle \alpha_A} \\ A^\perp \parallel A & \xleftarrow{l_A^\perp \parallel l_A} & \widehat{A}^\perp \parallel \widehat{A} & \xrightarrow{r_A^\perp \parallel r_A} & A^\perp \parallel A \end{array}$$

6.2.2. *Composition of span-strategies.* The following lemma permits lifting composition of essential strategies to composition of span-strategies:

LEMMA 3.47. *Consider two commuting diagrams between maps:*

$$\begin{array}{ccc} S_1 & \xrightarrow{f} & S_2 \\ \downarrow{\scriptstyle \sigma_1} & & \downarrow{\scriptstyle \sigma_2} \\ A_1^\perp \parallel B_1 & \xrightarrow{h_1^\perp \parallel h_2} & A_2^\perp \parallel B_2 \end{array} \qquad \begin{array}{ccc} T_1 & \xrightarrow{g} & T_2 \\ \downarrow{\scriptstyle \tau_1} & & \downarrow{\scriptstyle \tau_2} \\ B_1^\perp \parallel C_1 & \xrightarrow{h_2^\perp \parallel h_3} & B_2^\perp \parallel C_2 \end{array}$$

where f and g are assumed to preserve incompatible extensions. Then, there is a map $g \circledcirc f : T_1 \circledcirc S_1 \to T_2 \circledcirc S_2$ such that the following diagram commutes:

$$\begin{array}{ccc} T_1 \odot S_1 & \xrightarrow{g \odot f} & T_2 \odot S_2 \\ {\scriptstyle \tau_1 \odot \sigma_1}\downarrow & & \downarrow{\scriptstyle \tau_2 \odot \sigma_2} \\ A_1^\perp \parallel C_1 & \xrightarrow{h_1^\perp \parallel h_3} & A_2^\perp \parallel C_2 \end{array}$$

and $g \odot f$ preserves incompatible extensions.

Note that horizontal maps are *total* as they correspond to relabelling, while vertical maps are pre-strategies and hence partial.

PROOF. Define $g \circledast f : T_1 \circledast S_1 \rightharpoonup T_2 \circledast S_2$ by the universal property (Lemma 2.47) as $\langle (f \parallel h_3) \circ \Pi_1, (h_1 \parallel g) \circ \Pi_2 \rangle$ so that the following diagram commutes:

$$\begin{array}{ccc} T_1 \circledast S_1 & \xrightarrow{g \circledast f} & T_2 \circledast S_2 \\ {\scriptstyle \tau_1 \circledast \sigma_1}\downarrow & & \downarrow{\scriptstyle \tau_2 \circledast \sigma_2} \\ A_1^\perp \parallel C_1 & \xrightarrow{h_1^\perp \parallel h_3} & A_2^\perp \parallel C_2 \end{array}$$

To conclude we need to show that $g \circledast f$ indeed restricts to a map $T_1 \odot S_1 \to T_2 \odot S_2$. It is clear that $g \circledast f$ preserves visible events (by the previous diagram). We show that $g \circledast f$ preserves incompatible extensions, which in turn implies that it preserves essential events.

Let $x \in \mathscr{C}(T_1 \circledast S_1)$ with p and p' two incompatible extensions. By Lemma 2.45 and the fact that σ_1 and τ_1 are essential strategies that either $\Pi_1 p$ and $\Pi_1 p'$ are both in S or $\Pi_2 p$ and $\Pi_2 p'$ are in T. Assume the former case for instance.

By using the fact that f preserves incompatible extensions we know that $f(\Pi_1 p)$ and $f(\Pi_1 p')$ are incompatible extensions of $f(\Pi_1 x)$. From there, it follows $\Pi_1((g \circledast f)p)$ and $\Pi_1((g \circledast f)p')$ are incompatible extensions of $\Pi_1((g \circledast f)x)$ which in turn, implies that $(g \circledast f)p$ and $(g \circledast f)p'$ are incompatible extensions of $(g \circledast f)x$.

Since $g \circledast f$ preserves essential events, it restricts to $g \odot f : T_1 \odot S_1 \to T_2 \odot S_2$ that still preserves incompatible extensions between nonnegative events. □

Thus, from two span-strategies

$$\begin{array}{ccccc} S & \xleftarrow{l_S} & \widehat{S} & \xrightarrow{r_S} & S \\ \downarrow{\scriptstyle \sigma} & & \downarrow{\scriptstyle \widehat{\sigma}} & & \downarrow{\scriptstyle \sigma} \\ A^\perp \parallel B & \xleftarrow{l_A \parallel l_B} & \widehat{A}^\perp \parallel \widehat{B} & \xrightarrow{r_A \parallel r_B} & A^\perp \parallel B \end{array}$$

$$\begin{array}{ccccc} T & \xleftarrow{l_T} & \widehat{T} & \xrightarrow{r_T} & T \\ \downarrow{\scriptstyle \tau} & & \downarrow{\scriptstyle \widehat{\tau}} & & \downarrow{\scriptstyle \tau} \\ B^\perp \parallel C & \xleftarrow{l_B^\perp \parallel l_C} & \widehat{B}^\perp \parallel \widehat{C} & \xrightarrow{r_B^\perp \parallel r_C} & B^\perp \parallel C \end{array}$$

we obtain component-wise a new span-strategy:

$$\begin{array}{ccccc} T \circledcirc S & \xleftarrow{l_T \circledcirc l_S} & \widehat{T} \circledcirc \widehat{S} & \xrightarrow{r_T \circledcirc r_S} & T \circledcirc S \\ \downarrow{\scriptstyle \tau \circledcirc \sigma} & & \downarrow{\scriptstyle \widehat{\tau} \circledcirc \widehat{\sigma}} & & \downarrow{\scriptstyle \tau \circledcirc \sigma} \\ A^\perp \parr C & \xleftarrow{l_A^\perp \parr l_C} & \widehat{A}^\perp \parr \widehat{C} & \xrightarrow{r_A^\perp \parr r_C} & A^\perp \parr C \end{array}$$

Together, we expect composition of span-strategies and the copycat span-strategy to form a category. As for CG, it can be shown to be a bicategory (see **[CCW14]** for more details). Isomorphisms of span-strategies are defined in the obvious way: $(\sigma, \widehat{\sigma}) \cong (\tau, \widehat{\tau})$ when there exist isomorphisms $\varphi : S \cong T$ and $\widehat{\varphi} : \widehat{S} \cong \widehat{T}$ commuting with the projections $\varphi \circ l_S = l_T \circ \widehat{\varphi}$ and similarly for the right projection.

By lifting the structure from CG, we get the following proposition:

PROPOSITION 3.48. *The following data forms a category* SpanCG*:*

- Objects*: span-games*
- Morphisms from $\mathbb{A}$ to $\mathbb{B}$ *span-strategies from $\mathbb{A}$ to $\mathbb{B}$ up to isomorphism*
- Copycat and composition*: as described above.*

6.2.3. *Compact-closure of* SpanCG. Compact-closure of SpanCG is established as for CG, by lifting structural morphisms from the category of span of event structures to that of span-games and strategies.

A map of spans $(f, \widehat{f}) : \mathbb{A} \to \mathbb{B}$ such that both components are receptive and courteous, as in Chapter 2, can be lifted to a strategy $\overline{(f, \widehat{f})}$ obtained by composition in $\mathrm{Span}(\mathcal{E})$:

$$\begin{array}{ccccc} \mathbb{C}_A & \xleftarrow{\varpi_{l_A}} & \mathbb{C}_{\widehat{A}} & \xrightarrow{\varpi_{r_A}} & \mathbb{C}_A \\ \downarrow{\scriptstyle \varpi_A} & & \downarrow{\scriptstyle \varpi_{\widehat{A}}} & & \downarrow{\scriptstyle \varpi_A} \\ A^\perp \parr A & \xleftarrow{l_A^\perp \parr l_A} & \widehat{A}^\perp \parr \widehat{A} & \xrightarrow{r_A^\perp \parr r_A} & A^\perp \parr A \\ \downarrow{\scriptstyle A^\perp \parr f} & & \downarrow{\scriptstyle \widehat{A}^\perp \parr \widehat{f}} & & \downarrow{\scriptstyle A^\perp \parr f} \\ A^\perp \parr B & \xleftarrow{l_A^\perp \parr l_B} & \widehat{A}^\perp \parr \widehat{B} & \xrightarrow{r_A^\perp \parr r_B} & A^\perp \parr B \end{array}$$

In other words, it is defined as the span morphism with components $\overline{f}$ and $\overline{\widehat{f}}$, where $\overline{-}$ denotes the lifting operation of **[CCRW]**.

By the same argument as in Chapter 2, structural morphisms for the symmetric monoidal structure of $\mathrm{Span}(\mathcal{E})$ can be lifted to SpanCG. The equations for a compact-closed category can be easily deduced from CG (the fact that projections are preserved is a routine check.)

6.3. Embedding of $\sim$-tCG$^{\cong}_{\circledcirc}$. Our original goal was to show that $\sim$-tCG$^{\cong}_{\circledcirc}$ is a compact-closed category. To do so, we will embed it into $\mathrm{Span}(\mathrm{CG})$ and deduce the equations from the embedding.

LEMMA 3.49. *We have the following:*

(1) Let $\mathcal{A}$ be a tcg. The tuple $(A, Pr(\tilde{A}), l_A, r_A)$ is a span-game.

(2) Let $\mathcal{A}, \mathcal{B}$ be tcgs and $\sigma : \mathcal{S} \nrightarrow \mathcal{A}^\perp \parr \mathcal{B}$ a $\sim$-strategy. Then the preservation of symmetry of σ induces a map $\widehat{\sigma} : Pr(S) \to Pr(A)^\perp \parr Pr(\mathcal{B})$ which makes $(\sigma, \widehat{\sigma})$ a span-strategy from $(A, Pr(A), l_A, r_A)$ to $(B, Pr(B), l_B, r_B)$.

PROOF. *(1).* The first item is routine: in particular, the race-freeness of $\mathrm{Pr}(\tilde{A})$ is an immediate consequence of Lemma 3.37.

(2). The action of $\widehat{\sigma}$ on a prime $\theta : [s] \cong [s']$ is given by the restriction of $\sigma\theta : \sigma[s] \cong \sigma[s']$ (which exists because σ preserves symmetry) to $[\sigma s]$. The only claim left to prove is that $(S, \mathrm{Pr}(\tilde{S}), l_S, r_S)$ is thin. Assume incompatible extensions in $\mathrm{Pr}(\tilde{S})$: let $z \in \mathscr{C}(\mathrm{Pr}(\tilde{S}))$ that can be extended incompatibly by two nonnegative $\theta_1 : [s_1] \cong [s_1'], \theta_2 : [s_2] \cong [s_2'] \in \mathrm{Pr}(\tilde{S})$. By the isomorphism $\mathscr{C}(\mathrm{Pr}(\tilde{E})) \cong \tilde{E}$, z corresponds to a bijection $\theta : x \cong y$. We prove that $x \cup \{s_1, s_2\} = l_S(z \cup \{\theta_1, \theta_2\}) \notin \mathscr{C}(S)$ thus establishing that l_S preserves nonnegative incompatible extensions.

Assume that $x \cup \{s_1, s_2\}$ is a configuration. Then $\theta \cup \theta_1$ must extend by a pair (s_2, s_2'') to θ_1'. Hence $\theta_1' \restriction_{[s_2]} \circ \theta_2^{-1} : [s_2'] \cong [s_2'']$ is a valid bijection of $\tilde{S}$ that extends the identity on $[s_2) \subseteq x$ by nonnegative (s_2', s_2''). Since $\tilde{S}$ is thin, $s_2' = s_2''$. This implies that $\theta \cup \theta_1 \cup \{(s_1, s_2)\} = \theta \cup \theta_1 \cup \theta_2$ is a valid bijection in $\tilde{S}$, directly contradicting the fact that θ_1 and θ_2 are incompatible extensions of z. □

Having a way to send tcgs and $\sim$-strategies to their span counterparts, we now prove that this map is functorial. For copycat it follows from this lemma proved in **[CCW14]**:

LEMMA 3.50. *Let $\mathcal{A}$ be a tcg. Write $\widehat{A} = Pr(\tilde{A})$. Then, there is an iso making the following diagram commute*

$$\begin{array}{ccccc} & & \mathbb{C}_A & & \\ & {\scriptstyle l_{\mathbb{C}_A}} \nearrow & & \nwarrow {\scriptstyle \mathbb{C}_{l_A}} & \\ Pr(\mathbb{C}_{\tilde{A}}) & & \overset{\cong}{\longleftrightarrow} & & \mathbb{C}_{\widehat{A}} \\ & {\scriptstyle r_{\mathbb{C}_A}} \searrow & & \swarrow {\scriptstyle \mathbb{C}_{r_A}} & \\ & & \mathbb{C}_A & & \end{array}$$

and which also preserves the projections to the span-game

$$A^\perp \parr A \xleftarrow{l_A} \widehat{A}^\perp \parr \widehat{A} \xrightarrow{r_A} A^\perp \parr A$$

In particular, this yields an isomorphism of span-strategies.

To relate both compositions, a bit more work is needed:

LEMMA 3.51. *Let $\sigma : \mathcal{S} \nrightarrow \mathcal{A}^\perp \parr \mathcal{B}$ and $\tau : \mathcal{T} \nrightarrow \mathcal{B}^\perp \parr \mathcal{C}$ be $\sim$-strategies. Write $\widehat{S}$ for $Pr(\tilde{S})$ and $\widehat{T}$ for $Pr(\tilde{T})$. There is an isomorphism such that the following commutes:*

$$\begin{array}{ccccc} & & T \circledcirc S & & \\ & {\scriptstyle l_{T \circledcirc S}} \nearrow & & \nwarrow {\scriptstyle l_T \circledcirc l_S} & \\ Pr(\tilde{T} \circledcirc \tilde{S}) & & \overset{\cong}{\longleftrightarrow} & & \widehat{T} \circledcirc \widehat{S} \\ & {\scriptstyle r_{T \circledcirc S}} \searrow & & \swarrow {\scriptstyle r_T \circledcirc r_T} & \\ & & \widehat{T} \circledcirc \widehat{S} & & \end{array}$$

and which also preserves the projections to the underlying span-game. In particular, this yields an isomorphism of span-strategies.

PROOF. *Interaction.* We first establish the commutation of the corresponding diagram for the interaction:

$$\begin{array}{ccccc} & & T \circledast S & & \\ & {\scriptstyle l_{T\circledcirc S}}\nearrow & & \nwarrow{\scriptstyle l_T \circledcirc l_S} & \\ \Pr(\tilde{T} \circledast \tilde{S}) & \xleftrightarrow{\cong} & & \xrightarrow{} & \widehat{T} \circledast \widehat{S} \\ & {\scriptstyle r_{T\circledcirc S}}\searrow & & \swarrow{\scriptstyle r_T \circledcirc r_T} & \\ & & \tilde{T} \circledast S & & \end{array}$$

As above, configurations of $\Pr(\tilde{T} \circledast \tilde{S})$ correspond canonically to symmetries in $\tilde{T} \circledast \tilde{S}$. By definition (above Lemma 3.25), those correspond to commuting squares between composite secured bijections $x \simeq y$ and $x' \simeq y'$

$$\begin{array}{ccccccc} x & \overset{\sigma}{\simeq} & \sigma x & = & \tau y & \overset{\tau}{\simeq} & y \\ {\scriptstyle \theta_A}\wr\!\!\mid\!\!\mid_{\scriptstyle S} & & & & & & {\scriptstyle \theta_T}\wr\!\!\mid\!\!\mid_{\scriptstyle \tilde{T}} \\ x' & \overset{\sigma}{\simeq} & \sigma x' & = & \tau y' & \overset{\tau}{\simeq} & y' \end{array}$$

In particular, this gives a bijection between pairs $(s, s') \in \theta_S$ and $(t, t') \in \theta_T$. This bijection is secured, since the upper and lower sides of the diagram are secured by hypothesis and θ_S and θ_T are order-isos. The bijections θ_S and θ_T canonically represent configurations $z_S \in \mathscr{C}(\widehat{S})$ and $z_T \in \mathscr{C}(\widehat{T})$ – so overall, diagrams as above canonically correspond to secured composite bijections between z_S and z_T, as required.

Hiding. An event $e \in \Pr(\tilde{T} \circledcirc \tilde{S})$ represents a bijection $\theta : [p_1] \cong [p_2]$ in the isomorphism family $\tilde{T} \circledcirc \tilde{S}$. By Lemma 3.28, we know it induces a unique witness $\theta' : [p_1]_{T\circledast S} \cong [p_2]_{T\circledast S}$. By the previous point this witness corresponds to an event $e' \in \tilde{T} \circledast \tilde{S}$. If e is visible, since this isomorphism preserves the projection on the game, so is e'. Moreover, if e is essential, so is e' because of the following triangle:

$$\begin{array}{ccccc} & & T \circledast S & & \\ & {\scriptstyle l_{T\circledcirc S}}\nearrow & & \nwarrow{\scriptstyle l_T \circledcirc l_S} & \\ \Pr(\tilde{T} \circledast \tilde{S}) & \cong & & & \widehat{T} \circledast \widehat{S} \end{array}$$

and the projections preserve (and reflect as maps of event structures) incompatible extensions. So the isomorphism on the interaction restricts to the hiding. □

From this result and Proposition 3.48 it follows that $\sim$-tCG$^{\cong}_{\circledcirc}$ is indeed a category up to isomorphism – hence a category up to weak isomorphism. Moreover, there is a functor $\sim$-tCG$^{\cong}_{\circledcirc} \to$ SpanCG which is faithful (up to isomorphism).

Finally to deduce the compact-closure of $\sim$-tCG$^{\cong}_{\circledcirc}$ we use the same technique as in Chapter 2 to lift the structural morphisms of the symmetric monoidal category $\mathcal{E}$ to $\sim$-tCG$^{\cong}_{\circledcirc}$:

DEFINITION 3.52. Let $f : \mathcal{A} \to \mathcal{B}$ be a strong-receptive, courteous map of essps. Then its **lifting** is the $\sim$-strategy

$$\overline{f} = (\mathcal{A}^{\perp} \parallel f) \circ \sigma : \mathbb{C}_{\mathcal{A}} \to \mathcal{A}^{\perp} \parallel \mathcal{B}$$

which is a morphism from $\mathcal{A}$ to $\mathcal{B}$ in $\sim$-tCG$^{\cong}_{\circledcirc}$ (in particular, it is thin). Likewise, if $f : \mathcal{B}^{\perp} \to \mathcal{A}^{\perp}$ is receptive and courteous, we define its **co-lifting**:

$$\begin{array}{rcccl} \overline{f}^{\perp} & : & \mathbb{C}_B & \to & \mathcal{A}^{\perp} \parallel \mathcal{B} \\ & & c & \mapsto & (f \parallel B) \circ \varkappa_B(c) \end{array}$$

This lifting enjoys the usual property: composition with a lifted map amounts to relabelling:

LEMMA 3.53. *Let $f : \mathcal{B} \to \mathcal{C}$ be a receptive courteous map of event structures with symmetry and polarities, and $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}^{\perp} \parallel \mathcal{B}$ be a $\sim$-strategy. Then, there is an isomorphism of strategies*

$$\overline{f} \circledcirc \sigma \cong (\mathcal{A}^{\perp} \parallel f) \circ \sigma$$

Likewise, for $f : \mathcal{B}^{\perp} \to \mathcal{A}^{\perp}$ receptive courteous and $\sigma : \mathcal{S} \rightharpoonup \mathcal{B}^{\perp} \parallel \mathcal{C}$ an essential strategy, there is an isomorphism of strategies:

$$\sigma \circledcirc \overline{f} \cong (f \parallel \mathcal{C}) \circ \sigma$$

PROOF. Using the embedding as span-strategies, we can apply Lemma 2.89 on both components. □

Since this lifting is compatible with that of SpanCG, we can use this lifting to import the structural morphisms from $\mathcal{E}^{\sim}$ and deduce the equational laws from SpanCG, yielding:

PROPOSITION 3.54. *$\sim$-tCG$^{\cong}_{\circledcirc}$ is compact-closed.*

PROOF. The proof structure is the same as for CG$^{\cong}_{\circledcirc}$: associativity follows from embedding in SpanCG. The symmetric monoidal structure is obtained by lifting that of $\mathcal{E}^{\sim}$. Finally, the compact-closed structure is the natural extension of that of CG$^{\cong}_{\circledcirc}$: the dual of a tcg $\mathcal{A}$ is $\mathcal{A}^{\perp}$ and the strategies ϵ_A and η_A extend to $\sim$-strategies:

$$\begin{array}{rcccl} \eta_A & : & \mathbb{C}_{\mathcal{A}} & \to & 1^{\perp} \parallel (\mathcal{A}^{\perp} \parallel \mathcal{A}) \\ \epsilon_A & : & \mathbb{C}_{\mathcal{A}} & \to & (\mathcal{A} \parallel \mathcal{A}^{\perp})^{\perp} \parallel 1 \end{array} \quad \square$$

In the next chapter, we will see how to carve a cartesian-closed category within $\sim$-tCG$^{\cong}_{\circledcirc}$ by using duplicated arenas.

CHAPTER 4

Concurrent Hyland-Ong games

In this chapter, we build a cartesian-closed category CHO arising as a subcategory of $\sim$-tCG$^{\cong}_{\circledcirc}$. As such, it supports an interpretation of nonlinear higher-order computation that will be central to the next chapters. Nonlinearity will be accommodated by the ! construction presented in Section 1 of Chapter 3.

As a first example of the expressivity of this model, we show it supports a concurrent interpretation of the nondeterministic λ-calculus that is adequate for must-equivalence, result that relies on essential events to track hidden divergences.

Related work. The first work exploring nondeterministic games models adequate for must convergence was [**HM99**], representing the hidden divergences as *stopping traces*. Such an approach was replicated in our setting via *stopping configurations* [**CHLW14**]. Both suffer the same drawback: even though they capture more behaviours, these models are tailored for must convergence. For instance, Harmer's model is not sound for fair convergence. Our method, using essential events, allows to prove a very strong link between the operational semantics and the denotational semantics (Theorem 4.29) from which adequacy for a variety of notions of convergences can be derived.

Another line of work with similar goals is that of Hirschowitz *et al.* [**Hir13, EHS15**] that provides models capturing fair testing. Their model has similarities with ours, in particular it also records *all* internal events. However, they do not investigate hiding or composition in their models.

Outline of the chapter. In Section 1, we recall the exponential-like operation introduced in Section 1 (Chapter 3) and demonstrate informally how to interpret nondeterministic λ-terms in this setting. In section 2, we present the cartesian-closed category CHO. In Section 3, we properly define the interpretation of nondeterministic λ-terms inside CHO and show that it is adequate for may and must convergences, for nondeterministic PCF.

Contributions of this chapter. The construction of the cartesian-closed category in the setting without essential events is joint work with Pierre Clairambault. This chapter presents a generalization of the construction of CHO to the framework using essential events, introduced in Chapter 2.

1. Nonlinear nondeterministic strategies

Given an arena A (ie. an alternating forest – see Definition 3.1), we have seen how to expand A into a new arena $!A$ such that strategies on $!A$ can be seen as nonlinear strategies on A. In this section, we illustrate how this can be used to interpret terms from nondeterministic PCF, by providing examples of terms and their desired interpretation.

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \Rightarrow B} \qquad \frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}$$

$$\frac{}{\Gamma \vdash \mathcal{Y} : (A \Rightarrow A) \Rightarrow A} \qquad \frac{}{\Gamma \vdash \mathrm{tt} : \mathbb{B}} \qquad \frac{}{\Gamma \vdash \mathrm{ff} : \mathbb{B}} \qquad \frac{}{\Gamma \vdash \mathtt{choice} : \mathbb{B}} \qquad \frac{}{\Gamma \vdash \underline{n} : \mathbb{N}}$$

$$\frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \mathrm{succ}\, M : \mathbb{N}} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \mathrm{pred}\, M : \mathbb{N}} \qquad \frac{\Gamma \vdash M : \mathbb{N}}{\Gamma \vdash \mathrm{null}\, M : \mathbb{B}} \qquad \frac{\Gamma \vdash M : \mathbb{B} \qquad \Gamma \vdash N_i : \mathbb{X}}{\Gamma \vdash \mathtt{if}\, M\, N_1\, N_2 : \mathbb{X}}$$

FIGURE 1. Typing rules of **ndPCF**

1.1. Syntax and operational semantics of ndPCF.

1.1.1. *Nondeterministic PCF.* In this chapter and the following ones, we consider a variant of PCF enhanced with a nondeterministic boolean `choice`. The resulting language, **ndPCF**, contains the simply-typed λ-calculus, a fixpoint combinator $\mathcal{Y}$ and primitives to manipulate base types (booleans or integers):

$$\begin{array}{llll} \text{(Types)} & A, B & ::= & \mathbb{B} \mid \mathbb{N} \mid A \Rightarrow B \\ \text{(Terms)} & M, N & ::= & x \mid \lambda x.\, M \mid M\, N \mid \mathcal{Y} \\ & & & \mid \mathrm{tt} \mid \mathrm{ff} \mid \mathtt{choice} \mid \mathtt{if}\, M\, N_1\, N_2 \\ & & & \mid \underline{n} \mid \mathrm{succ}\, M \mid \mathrm{pred}\, M \mid \mathrm{null}\, M \end{array}$$

A type is **ground** when it is $\mathbb{B}$ or $\mathbb{N}$. We will use $\mathbb{X}, \mathbb{Y}, \ldots$ to range over ground types. Typing rules are standard (with the exception that conditional branches are restricted to be of ground type) and given in Figure 1. Conditionals at higher-order type can be expanded to fit this presentation (eg. $\mathtt{if}\, b\, f\, g$ would be syntactic sugar for the term $\lambda x.\, \mathtt{if}\, b\, (f\, x)\, (g\, x)$).

As usual we write $\bot$ for the diverging term $\mathcal{Y}\, (\lambda x.\, x)$. We also write $M; N$ as a short-hand for $\mathtt{if}\, M\, N\, N$: evaluate M, discard its result and then continue to N.

1.1.2. *Operational semantics.* Figure 2 defines the small-step operational semantics for weak head reduction of **ndPCF** terms. Write $\to^*$ for the transitive and reflexive closure of $\to$. A **value** is a term of the form tt, ff or $\underline{n}$. Values are the only terms of ground type that cannot be reduced any further (such terms are called **normal form**):

LEMMA 4.1 (Progress). *Any closed normal form of ground type is a value.*

PROOF. By induction on a closed normal form M of ground type. □

1.1.3. *Notions of convergences.* Since **ndPCF** is non-deterministic, there is no canonical notion of convergence on which a testing equivalence can be based. In this thesis, we consider two of the most common notions:

- $M \Downarrow_{\mathrm{may}}$: M **may converge** whenever $M \to^* v$ for some $v \in \{\mathrm{tt}, \mathrm{ff}, \underline{0}, \underline{1}, \ldots\}$.
- $M \Downarrow_{\mathrm{must}}$: M **must converge** whenever M has no infinite reductions.

Must-convergence implies may-convergence. The converse does not hold, as witnessed by $M = \mathtt{if}\, \mathtt{choice}\, \mathrm{tt}\, \bot$ that may converge to tt but must not converge because of the infinite reduction sequence $M \to^* \bot \to \ldots$.

Both notions of convergence induce an observational equivalence, obtained as a closure under context. A **context** for type $\Gamma \vdash A$ is a closed and normal term $\mathcal{C}[]$ with a hole such that for all term $\Gamma \vdash M : A, \vdash \mathcal{C}[M] : \mathbb{B}$. Two terms $\Gamma \vdash M, N : A$ are **may-equivalent** (written $M \simeq_{\mathrm{may}} N$) if for all context $\mathcal{C}[]$ for type $\Gamma \vdash A$,

$$\frac{}{\texttt{choice} \to \mathsf{tt}}\ \texttt{Choice-true} \qquad \frac{}{\texttt{choice} \to \mathsf{ff}}\ \texttt{Choice-false}$$

$$\frac{M \to M'}{\texttt{if}\, M\, N_1\, N_2 \to \texttt{if}\, M'\, N_1\, N_2}\ \texttt{Cong-if} \qquad \frac{}{\texttt{if}\, \mathsf{tt}\, N_1\, N_2 \to N_1}\ \texttt{If-true}$$

$$\frac{}{\texttt{if}\, \mathsf{ff}\, N_1\, N_2 \to N_2}\ \texttt{If-false} \qquad \frac{}{\mathrm{succ}\, \underline{n} \to \underline{n+1}}\ \texttt{Succ-n}$$

$$\frac{}{\mathrm{pred}\, \underline{0} \to \underline{0}}\ \texttt{Pred-zero} \qquad \frac{}{\mathrm{pred}\, \underline{n+1} \to \underline{n}}\ \texttt{Pred-succ} \qquad \frac{}{\mathrm{null}\, \underline{0} \to \mathsf{tt}}\ \texttt{Null-zero}$$

$$\frac{}{\mathrm{null}\, \underline{n+1} \to \mathsf{ff}}\ \texttt{Null-succ} \qquad \frac{M \to M'}{\mathrm{succ}\, M \to \mathrm{succ}\, M'}\ \texttt{Cong-succ}$$

$$\frac{M \to M'}{\mathrm{null}\, M \to \mathrm{null}\, M'}\ \texttt{Cong-null} \qquad \frac{}{\mathcal{Y}\, M \to M\, (\mathcal{Y} M)}\ \texttt{Fixpoint}$$

$$\frac{}{(\lambda x.\, M) N \to M[N/x]}\ \texttt{Beta} \qquad \frac{M \to M'}{M\, N \to M'\, N}\ \texttt{Cong-app}$$

FIGURE 2. Weak head reduction for **ndPCF**

$\mathcal{C}[M] \Downarrow_{\text{may}}$ if and only if $\mathcal{C}[N] \Downarrow_{\text{may}}$. Two terms $\Gamma \vdash M, N : A$ are **must-equivalent** if for all context $\mathcal{C}$ for type $\Gamma \vdash A$, $\mathcal{C}[M] \Downarrow_{\text{must}}$ if and only if $\mathcal{C}[N] \Downarrow_{\text{must}}$.

Even though may convergence is weaker than must convergence, the induced equivalence relations are incomparable. Indeed, consider $M = \lambda b.\, \texttt{if}\, \texttt{choice}\, b\, \bot$ and $N = \lambda b.\, \bot$. By induction on contexts, it easy to see that M and N are must-equivalent. However $\mathcal{C}[] = []\, \mathsf{tt}$ is enough to distinguish M and N up to may equivalence. As a result, instead of must equivalence, we will be more interested in models well-behaved for may *and* must equivalence ($M \simeq_{\text{m\&m}} N$ whenever M and N are may *and* must equivalent).

1.2. Informal interpretation into $\mathrm{CHO}_{\circledcirc}$. In the rest of the section, we investigate informally what an interpretation of this language could look like using $\sim$-$\mathrm{tCG}_{\circledcirc}$ as the framework, and using the expansion operation ! defined in the previous chapter. The idea is to interpret types as arenas and terms as strategies playing on the corresponding expanded arenas.

1.2.1. *Negativity and call-by-name.* To be a model of the call-by-name λ-calculus, our model needs to satisfy the law:

$$[\![(\lambda x.M)N]\!] = [\![M]\!] \qquad (x \text{ does not appear free in } M)$$

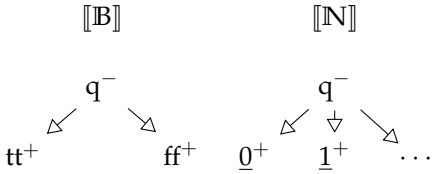
This amounts to having a certain strategy $e_A \in \sim\text{-}\mathrm{tCG}_{\circledcirc}(A, 1)$ which is natural in the sense that for all $\sigma \in \sim\text{-}\mathrm{tCG}_{\circledcirc}(A, B)$, $e_B \circledcirc \sigma = e_A \in \sim\text{-}\mathrm{tCG}(A, 1)$. The natural candidate for this strategy is the minimal strategy on $A^\perp \parallel 1$ that only plays the minimal negative moves of $A^\perp$ (to be receptive). However, if games are allowed to have minimal events of different polarities (ie. they are *non-polarized*), then this equation cannot be satisfied, already in the setting without symmetry: consider $A = \ominus \oplus$ and σ playing on $A^\perp \parallel 1$:

$$\sigma : \qquad A^\perp \qquad \| \quad 1$$

$$\ominus \longrightarrow \oplus$$

Since σ is a strategy, we have $\text{æ}_1 \circledcirc \sigma \cong \sigma$. However, in this case $\text{æ}_1 = e_1$ hence $e_1 \circledcirc \sigma$ is distinct from e_A. This forces us to restrict ourselves to *polarized arenas*. Moreover, as we want to model call-by-name computation, we need **negative arenas**: arenas whose initial events are all negative. This issue is not specific to our approach: it is already present in classic games models for the call-by-name λ-calculus [**HO00, AJM00**]. Dually, positive arenas can be used to model the call-by-value λ-calculus [**HY97**]. Polarization does not help to ensure that e_A is natural if strategies are unconstrained though: later we will also require strategies to be negative.

1.2.2. *Interpretation of types as negative arenas.* We now briefly sketch how types are interpreted as arenas in our setting. The interpretations $[\![\mathbb{B}]\!]$ and $[\![\mathbb{N}]\!]$ are simply given by the arena with a single negative move (the question) which is initial, and as many positive moves depending on it as there are values (answers):



These arenas are indeed negative. Negative arenas are stable under parallel composition (which will be the categorical product of our CCC). However, negativity is not preserved by the operation $\cdot^\perp \parallel \cdot$: for instance, $\mathbb{B}^\perp \parallel \mathbb{B}$ has a minimal positive event. Hence, for the arrow type, we cannot simply let $[\![A \Rightarrow B]\!] = [\![A]\!]^\perp \parallel [\![B]\!]$ (as the compact-closed structure of $\sim$-tCG$^{\cong}_{\circledcirc}$ suggests). To solve this problem, we follow the standard approach in HO game semantics which is to use a specific construction on arenas that preserves negativity:

DEFINITION 4.2 (Arrow construction on arenas). Let A, B be two arenas. Their **arrow** is $A \Rightarrow B$, with the following components.

- *Events, and polarity.* Those of:

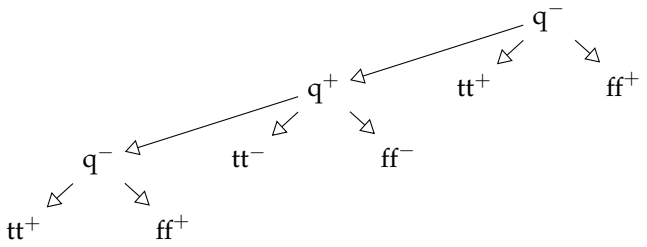
$$(\|_{b \in \min(B)} A^\perp) \parallel B$$

- *Causality.* Given by:

$$\leq_{(\|_{b \in \min(B)} A^\perp) \| B} \uplus \{((2, b), (1, (b, a))) \mid b \in \min(B) \;\&\; a \in A\}$$

where $\min(B)$ denotes the set of minimal events in B.

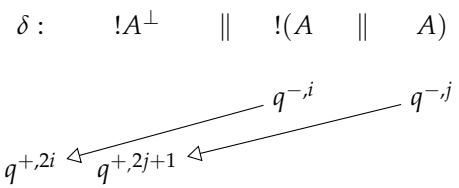
EXAMPLE 4.3. The arena $(\mathbb{B} \Rightarrow \mathbb{B}) \Rightarrow \mathbb{B}$ is:



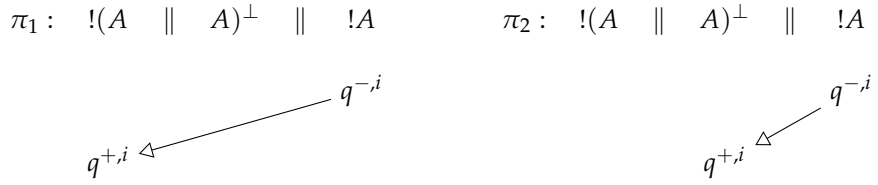
1.2.3. *Interpretation of terms as strategies.* Having defined the interpretation of types, we can look at the terms. First, if $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is a simply-typed context, then its interpretation is the arena $[\![\Gamma]\!] = [\![A_1]\!] \parallel \ldots \parallel [\![A_n]\!]$. Open terms $\Gamma \vdash t : A$ will be interpreted as certain strategies on the game $![\![\Gamma]\!]^\perp \parallel ![\![A]\!]$. In particular, this is not a Kleisli construction. Even though it is possible to perform a Kleisli construction on an exponential comonad inside $\sim$-tCG$^{\approx}_{\circledcirc}$, we follow this alternative choice here for historical reasons.

The exact interpretation will follow from the cartesian-closed structure of a category built out of $\sim$-tCG$^{\approx}_{\circledcirc}$ in Section 2. In the rest of this section, we present some examples of strategies arising from the interpretation. These examples are used to illustrate the phenomena at play and problems that we will encounter.

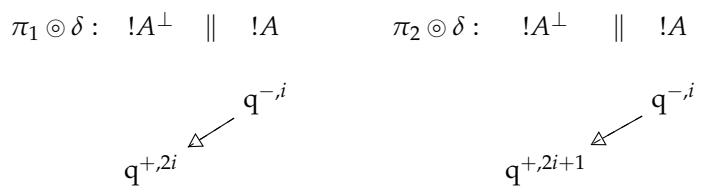
EXAMPLE 4.4 (Duplication and the need for symmetry). To illustrate the need for symmetry on expanded games, we look at the duplication δ from an arena A to $A \parallel A$, crucial to model non-linearity. This duplication should be a strategy on $!A^\perp \parallel !(A \parallel A)$ depicted for a type A interpreted by the singleton arena q^-:



To satisfy local injectivity, we have to pick different copy indices for the two positive moves using the fact that $!A \cong !A \parallel !A$ (isomorphism of event structures). Projections (terms: $x_1 : A, x_2 : A \vdash x_i : A$ for $i \in \{1, 2\}$) are represented by:



Naturally, when we pre-compose the two projections with δ we obtain the following two strategies:



They are not isomorphic in the sense of Chapter 2: when Opponent plays $\mathrm{q}^{-,0}$, then $\pi_1 \circledcirc \delta$ answers with copy index zero while $\pi_2 \circledcirc \delta$ answers with copy index one. They are however weakly isomorphic (in the sense of Chapter 3). Symmetry is here crucial to ensure (in particular) that the equation $\pi_1 \circledcirc \delta \cong \pi_2 \circledcirc \delta$ holds, necessary to model the call-by-name λ-calculus.

Note that the phenomenon described here will resurface in the interpretation of the simply-typed λ-calculus to show for instance that the β-equivalent terms $z : A \vdash (\lambda xy.x)zz : A$ and $z : A \vdash (\lambda xy.y)zz$ are mapped to weakly isomorphic strategies (but not isomorphic strategies).

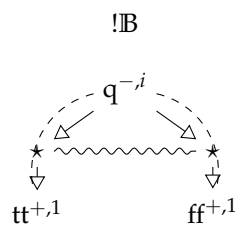
Next, we describe how our model represents higher-order computation:

EXAMPLE 4.5 (The Church integer 2). Consider the term $f : A \Rightarrow A, x : A \vdash f(fx) : A$. Figure 3 depicts the interpretation of f where A is interpreted by the arena $\mathbf{proc} = \mathbf{run}^- \multimap \mathbf{done}^+$.

The reader familiar with HO game semantics will notice that prime configurations (those with a top element) exactly correspond to P-views of the HO interpretation of the term: our semantics computes an expansion of the P-view tree in that example. (See Section 1 in Chapter 6 for more details)

Each $\mathbf{run}^+$ move corresponds to a variable call (leftmost f first, then rightmost f, then x in the causal order). Pointers of a $\mathbf{done}^+$ indicate the corresponding $\mathbf{run}^-$ it answers to. Copy indices are arbitrary and just here to ensure that the strategy is locally injective: for that purpose an injection $\langle\cdot\rangle : \mathbb{N}^* \to \mathbb{N}$ is used to encode indices of the *negative* moves appearing *after the justifier* of a positive move. We will see in Chapter 6 a way, for well-behaved (*innocent*) strategies, to reconstruct automatically this information.

EXAMPLE 4.6 (`choice` and branching point). In our model, `choice` can simply be interpreted by the following strategy:



Remember that strategies are not allowed to feature conflict between positive moves, hence the need for essential events.

Using this operator we can represent more complex nondeterministic computations, for instance: $x : \mathbb{B} \vdash \mathtt{if}\,\mathtt{choice}\,(\mathtt{if}\,x\,\perp\,\perp)\,x : \mathbb{B}$. This flips a coin: if it returns false, then it evaluates its argument x and returns the result, otherwise it

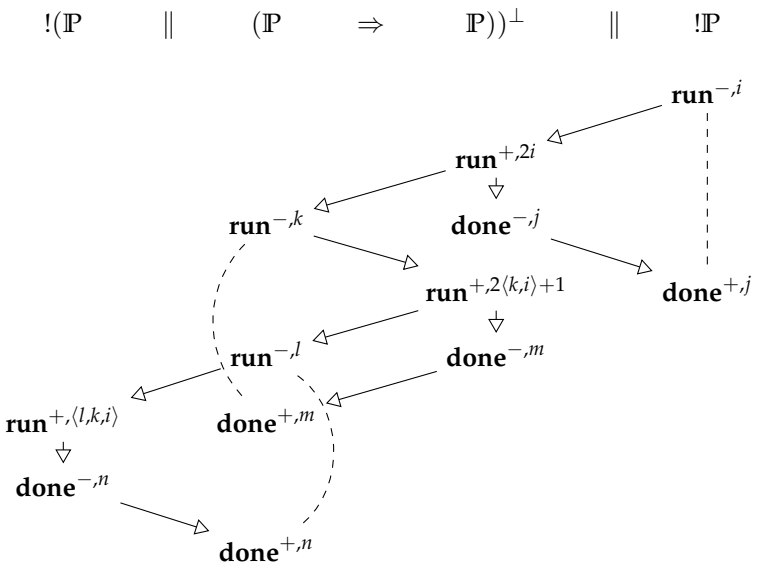


FIGURE 3. Interpretation of the church integer 2

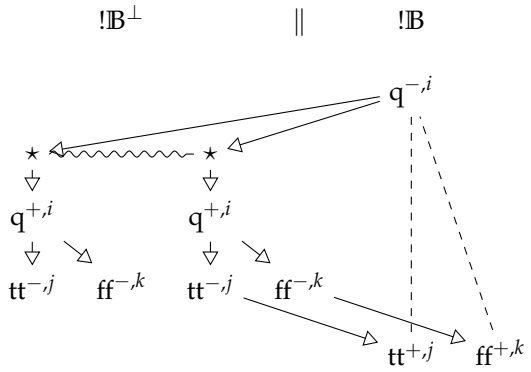


FIGURE 4. Retaining nondeterministic branching point

evaluates x and then diverges. The corresponding strategy, depicted in Figure 4 differs from the interpretation of $x : \mathbb{B} \vdash x$.

That is due to our semantics remembering the nondeterministic branching point (*before* the evaluation of x – when the coin is tossed). In this case, less intensional semantics (eg. traces) do not remember the nondeterminism at all since one behaviour is included in the other.

The interpretation of this term also differs from that of the term

$$x : \mathbb{B} \vdash \mathtt{if}\ x\ (\mathtt{if}\ \mathtt{choice}\ \mathsf{tt}\ \bot)\ (\mathtt{if}\ \mathtt{choice}\ \mathsf{ff}\ \bot)$$

where nondeterministic choice is delayed after the evaluation of x.

2. A cartesian-closed category

In this section, we prove that there is a certain subcategory of $\sim$-tCG$_{\circledcirc}$ which is cartesian-closed. Its objects will be *negative* arenas A and, its maps from A to B certain $\sim$-strategies on the game $!A^{\perp} \parallel !B$. To get a CCC, we need to impose further restrictions on $\sim$-strategies.

Note that in this section, a $\sim$-strategy on an arena A means a $\sim$-strategy on $!A$, and similarly a $\sim$-strategy from an arena A to B means a $\sim$-strategy on $!A^{\perp} \parallel !B$.

Binary conflict. The first restriction we impose, even though not technically necessary, is *binary conflict* (Definition 2.2), to get a more concrete grasp on strategies. The languages of interest here can all be interpreted with binary conflict.

Since our arenas and their expanded games are all conflict-free, copycat is always conflict-free and consequently has binary conflict. This condition is trivially preserved by parallel composition, and composition of strategies:

LEMMA 4.7. *Let $\sigma : S \rightharpoonup A^{\perp} \parallel B$ and $\tau : T \rightharpoonup B^{\perp} \parallel C$ be pre-strategies with binary conflict. Then $T \circledast S$, and $T \circledcirc S$ all have binary conflict.*

PROOF. We first show that the interaction $T \circledast S$ has binary conflict. Define $p \sharp_{T \circledast S} p'$ when $\Pi_1 p \sharp_{S \parallel C} \Pi_1 p'$ or $\Pi_2 p \sharp_{A \parallel T} \Pi_2 p'$. That $\sharp_{T \circledast S}$ indeed spans $\mathrm{Con}_{T \circledast S}$ is a consequence of Lemma 2.45.

Now to conclude, we simply remark that if S has binary conflict, then so does $S \downarrow V$ for any $V \subseteq S$ via $\sharp_{S \downarrow V} = \sharp_S \cap V^2$. □

Since binary conflict is preserved by all our constructions, from now on we only consider event structures with binary conflict. Similarly, for an event structure with symmetry $\mathcal{A}$ we always assume that A has binary conflict. Note that in general the event structure representing the symmetry, $\widehat{A}$ (as built in Section 6, Chapter 3), will not have binary conflict.

2.1. The cartesian structure. We start off by investigating which restrictions on strategies are necessary in order to get a cartesian category. The empty arena 1 is the candidate terminal object and parallel composition is the candidate product.

2.1.1. *Terminal object.* To make sure 1 is indeed terminal, we need to go further than in Section 1.2.1 and to cut down the space of strategies. Even though there is a unique $\sim$-strategy on the game $!1$ (the empty strategy), it is not the case in general that there is a unique $\sim$-strategy on $!A^{\perp} \parallel !1$ for a negative arena A. Since A is negative, $A^{\perp}$ does not have any negative minimal event, hence the empty strategy $e_A : \emptyset \to A^{\perp} \parallel 1$ is always a $\sim$-strategy. However, it might not be the only one. Indeed for $A = \ominus$, there is another strategy on $!A^{\perp} \parallel 1$ which simply plays the positive initial move of $A^{\perp}$ with an arbitrarily chosen copy index. To forbid this behaviour, we introduce negativity on the level of strategies as well: a $\sim$-strategy is **negative** when all its minimal events are negative.

LEMMA 4.8. *Let A be a negative arena. There is a unique negative $\sim$-strategy on the game $!A^{\perp} \parallel 1$.*

PROOF. The existence of such a strategy has already been established. Assume a negative $\sim$-strategy $\sigma : \mathcal{S} \to A^{\perp} \parallel 1$. If S is not empty, it has a minimal event $s \in S$ which is mapped to an event σs necessarily in $!A^{\perp}$, and necessarily minimal. By assumption s is negative but σs positive, absurd. □

From this result, we deduce that 1 is a terminal object in the category of negative arenas and negative strategies up to weak isomorphism – given that such a category is well-defined:

LEMMA 4.9. *We have the following:*

- *For any game A, $\mathbb{C}_A$ is negative,*
- *For any games A, B, C, if $\sigma : A \nrightarrow B$ and $\tau : B \nrightarrow C$ are composable negative strategies, then $\tau \odot \sigma$ is negative.*

As a result the category of negative arenas and negative $\sim$-strategies (up to isomorphism) exists and has a terminal object.

PROOF. The minimal events of $\mathbb{C}_A$ are always negative, regardless of the game A, since every positive move depends on its corresponding negative move in the dual component, and as a result cannot be minimal.

Consider negative $\sim$-strategies $\sigma : \mathcal{S} \rightharpoonup !A^\perp \parallel !B$ and $\tau : \mathcal{T} \rightharpoonup !B^\perp \parallel !C$. Let p be a minimal event of the composition $T \odot S$, and p' a minimal event of $[p]_{T \circledast S}$. As Π_1 and Π_2 are map of event structures, they preserve minimality. Moreover, either $\Pi_1 p'$ belongs to S or $\Pi_2 p'$ belongs to T. If $\Pi_1 p'$ belongs to S, it is minimal there hence negative and sent to C. In other terms, p' is visible and $p = p'$ which proves that p is negative as desired.

Otherwise, with the same reasoning $\Pi_2 p'$ belongs to T and is minimal there: it is a negative move sent to B. This implies that $\Pi_1 p'$ is also in S, but positive and also minimal, which is absurd. □

Remark that, in particular without restricting to negative arenas, there is a subcategory of negative strategies in CG. However, it is not cartesian as illustrated above.

2.1.2. *Projections.* To show that $\parallel$ actually defines a categorical product, the first thing to do is to define projections from $A \parallel B$ to A and B. Those strategies are obtained by co-lifting (Definition 3.52). Recall that if $f : \mathcal{A} \to \mathcal{B}$ is a courteous and receptive map of event structures with symmetry between games, it induces a strategy $\bar{f}$ from $\mathcal{B}^\perp$ to $\mathcal{A}^\perp$. To define the projections, it is thus enough to provide such maps $i_A : !A \to !(A \parallel B)$ and $i_B : !B \to !(A \parallel B)$.

We only detail the case for the left projection. The map i_A is defined as follows:

$$\begin{aligned} !A &\to !(A \parallel B) \\ (a, \alpha) &\mapsto \left((0, a), \begin{array}{c} [(0,a)] \to \mathbb{N} \\ (0, a') \mapsto \alpha(a') \end{array} \right) \end{aligned}$$

Checking that this is a map of essps which is courteous and receptive is routine. Since expansion and parallel composition commute with duality, we can define the projection as follows:

$$\pi_1 = \overline{i_{A^\perp}}^\perp \in \sim\text{-tCG}_\odot(!(A \parallel B), !A)$$
$$\pi_2 = \overline{i_{B^\perp}}^\perp \in \sim\text{-tCG}_\odot(!(A \parallel B), !B)$$

2.1.3. *Pairing.* Given strategies σ from A to B and τ from A to C, we now form their pairing $\langle \sigma, \tau \rangle$ from A to $B \parallel C$. First, we remark that if σ and τ have disjoint image in the game $!A$, then the pairing is easy to define.

LEMMA 4.10. *Let $\sigma : \mathcal{S} \rightharpoonup {!A^\perp} \parallel {!B}$ and $\tau : \mathcal{T} \rightharpoonup {!A^\perp} \parallel {!C}$ be negative $\sim$-strategies such that no event of $!A$ is played by both σ and τ.*

Then the following map defines a negative $\sim$-strategy $\langle\!\langle \sigma, \tau \rangle\!\rangle$ from A to $B \parallel C$:

$$\langle\!\langle \sigma, \tau \rangle\!\rangle = [(!A^\perp \parallel i_B) \circ \sigma, (!A^\perp \parallel i_C) \circ \tau] : \mathcal{S} \parallel \mathcal{T} \rightharpoonup {!A^\perp} \parallel !(B \parallel C)$$

where $[f, g]$ denotes the set-theoretic co-pairing (remember that the set of events of $S \parallel T$ is the set-theoretic coproduct of the set of events of S and T).

PROOF. First, the assumption on σ and τ being disjoint on $!A$ implies directly that $\langle\!\langle \sigma, \tau \rangle\!\rangle$ is locally injective. It is routine to check that it is actually a map of event structures with symmetry which is courteous.

Strong-receptivity requires more care and relies on the fact that arenas are forests. Assume $\theta : x_S \parallel x_T \cong y_S \parallel y_T \in \widetilde{S \parallel T}$ such that $\langle\!\langle \sigma, \tau \rangle\!\rangle$ can be extended by negative $(d_1, d_2) \in !A^\perp \parallel !(B \parallel C)$. The problem is to know whether to apply strong-receptivity of σ or τ. If d_1 and d_2 are minimal, then they live either in $!B$ or in $!C$ and we can apply the corresponding strong-receptivity of σ or τ.

Otherwise, because the game is a forest, there exist unique $c_1 \rightarrowtriangle d_1$ and $c_2 \rightarrowtriangle d_2$ with $(c_1, c_2) \in (\langle\!\langle \sigma, \tau \rangle\!\rangle)\theta$. The pair (c_1, c_2) originates either in σ or in τ and we can apply strong-receptivity of the corresponding strategy to conclude. □

Note that this pairing operation obviously commutes with forgetting essential events: $\langle\!\langle \sigma, \tau \rangle\!\rangle_\downarrow \cong \langle\!\langle \sigma_\downarrow, \tau_\downarrow \rangle\!\rangle$ (both having $\mathcal{S}_\downarrow \parallel \mathcal{T}_\downarrow$ as underlying event structure with symmetry). This pairing behaves well with respect to projections:

LEMMA 4.11. *Let $\sigma : \mathcal{S} \rightharpoonup {!A^\perp} \parallel {!B}$ and $\tau : \mathcal{T} \rightharpoonup {!A^\perp} \parallel {!C}$ be negative $\sim$-strategies as in Lemma 4.10. Then we have the following weak isomorphisms:*

$$\pi_1 \circledcirc \langle\!\langle \sigma, \tau \rangle\!\rangle \approxeq \sigma \qquad \pi_2 \circledcirc \langle\!\langle \sigma, \tau \rangle\!\rangle \approxeq \tau$$

PROOF. We only detail the projection on A. We prove that the interaction $\pi_1 \circledast \langle\!\langle \sigma, \tau \rangle\!\rangle$ is isomorphic to $\varpi_{!B} \circledast \sigma$, and that the obvious restriction to the visible parts $\pi_1 \circledcirc \langle\!\langle \sigma_\downarrow, \tau_\downarrow \rangle\!\rangle$ and $\varpi_{!B} \circledcirc \sigma_\downarrow$ commute with the projection on the game $!A \parallel !C$. We deduce the final isomorphism by composing with $\varpi_B \circledcirc \sigma \cong \sigma$ which comes from σ being a $\sim$-strategy.

By Lemma 2.50, it is enough to build an order-isomorphism between $\mathscr{C}(\pi_1 \circledast \langle\!\langle \sigma, \tau \rangle\!\rangle)$ and $\mathscr{C}(\varpi_{!B} \circledast \sigma)$. We recall that configurations of the later are isomorphic to pairs $(x, y) \in \mathscr{C}(S) \times \mathscr{C}(\mathbb{C}_{!B})$ with $y \sqsubseteq_{!B} \sigma_\downarrow x$.

Configurations of $\pi_1 \circledast \langle\!\langle \sigma, \tau \rangle\!\rangle$ correspond to secured bijections

$$(x_S \parallel x_T) \parallel x_B \simeq y_A \parallel (x_S \parallel x_T)_\star \parallel (y_B^1 \parallel y_B^2)$$

where $x_S \in \mathscr{C}(S), x_T \in \mathscr{C}(T), \langle\!\langle \sigma, \tau \rangle\!\rangle (x_S \parallel x_T) = y_A \parallel (\mathrm{i}_B\, y_B^1)$, and $y_B^1 \parallel y_B^2 \in \mathscr{C}(\mathbb{C}_{!B})$, and where the bijection is the unique such that the image of events through the labelings $\langle\!\langle \sigma, \tau \rangle\!\rangle \parallel !B$ and $!A \parallel \pi_1$ match. As a result, $\langle\!\langle \sigma, \tau \rangle\!\rangle (x_S \parallel x_T)$ does not reach $!C$. But any minimal events of x_T are negative by negativity of τ, and hence must be in $!C$ (since A is negative). Therefore, x_T is empty. Hence mapping the secured bijection to (x_S, y_B^2) which corresponds to a configuration of $\varpi_{!B} \circledast \sigma$ extends to a bijection between domains of configurations, that preserves symmetry, yielding the desired isomorphism. □

So, we know how to construct a pairing behaving well with projections, when the paired strategies happen to have a disjoint image on $!A$. However, for arbitrary $\sigma : \mathcal{S} \rightharpoonup !A^\perp \parr !B$ and $\tau : \mathcal{T} \rightharpoonup !A^\perp \parr !C$, there might in general be collisions: events $s \in S$ and $t \in T$ such that $\sigma s = \tau t$. In such a case, the co-pairing as above fails local injectivity, and therefore does not correspond to a strategy. Fortunately, we can *relabel* moves of $\mathcal{S}$ and $\mathcal{T}$, not changing their weak isomorphism class, to ensure that there are no such collisions. For that, we note that there are maps of event structures with symmetry

$$\iota_e : !A^\perp \to !A^\perp \qquad \iota_o : !A^\perp \to !A^\perp$$

such that $\iota_e \sim_{\widetilde{!A_+}} \iota_o \sim_{\widetilde{!A_+}} \mathrm{id}_{!A^\perp}$, but such that ι_e and ι_o have disjoint codomain. For definiteness, say that ι_e sends (necessarily positive) minimal events with copy index i to the same events with copy index $2i$, and preserves the copy index of other events. Likewise, ι_o could follow the injection $i \mapsto 2i + 1$. These maps preserve the index of negative events, so that $\iota_e \sim_{\widetilde{!A_+}} \iota_o \sim_{\widetilde{!A_+}} \mathrm{id}_{!A^\perp}$.

Given arbitrary $\sigma : \mathcal{S} \rightharpoonup !A^\perp \parr !B$ and $\tau : \mathcal{T} \rightharpoonup !A^\perp \parr !C$, define:

$$\sigma_e = (\iota_e \parr !B) \circ \sigma \qquad \tau_o = (\iota_o \parr !C) \circ \tau$$

From $\iota_e \sim_{\widetilde{!A_+}} \iota_o \sim_{\widetilde{!A_+}} \mathrm{id}_{!A^\perp}$ it is obvious that $\sigma \cong \sigma_e$ and $\tau \cong \tau_o$, but σ_e and τ_o now have disjoint codomains: σ_e (resp. τ_o) only reaches indexing functions in $!A$ whose index for minimal events is even (resp. odd). Therefore, using Lemma 4.10, we define the **pairing** of σ and τ:

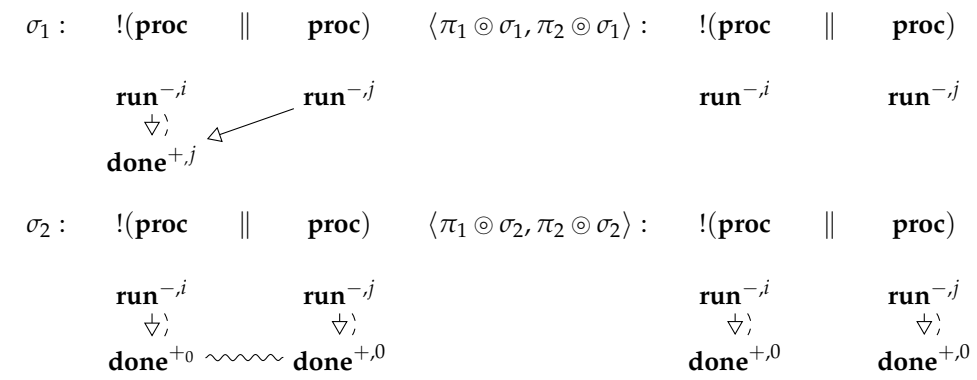
$$\langle \sigma, \tau \rangle = \langle\!\langle \sigma_e, \tau_o \rangle\!\rangle$$

We have, as required, $\pi_1 \circledcirc \langle \sigma, \tau \rangle = \pi_1 \circledcirc \langle\!\langle \sigma_e, \tau_o \rangle\!\rangle \cong \sigma_e \cong \sigma$, and for the same reason $\pi_2 \circledcirc \langle \sigma, \tau \rangle \cong \tau$. It is an immediate verification that $\langle -, - \rangle$ preserves weak isomorphism, so it will still make sense as an operation on the quotient category.

2.1.4. *Surjective pairing.* The last property we need in order to build a cartesian category is **surjective pairing**: for all negative $\sim$-strategies σ from A to $B \parr C$,

$$\sigma \cong \langle \pi_1 \circledcirc \sigma, \pi_2 \circledcirc \sigma \rangle.$$

Unfortunately, this property does not hold for negative strategies in general. Indeed, in $\langle \pi_1 \circledcirc \sigma, \pi_2 \circledcirc \sigma \rangle$, there cannot be any causal links (or conflicts) between what is played on B and on C: the two components are independent. Such a requirement is not imposed on σ that is free to have events depending on moves both from B and C, as in the following examples:



This operation $\sigma \mapsto \langle \pi_1 \circledcirc \sigma, \pi_2 \circledcirc \sigma \rangle$ removes interferences (causal dependence or conflicts) across components. To hope for surjective pairing, we need to forbid such behaviours, through the notion of **single-threaded** strategies:

DEFINITION 4.12. Let $\sigma : S \rightharpoonup A$ be a pre-strategy. We say that σ is **single-threaded** if it satisfies the following two conditions.

(1) For any $s \in S$, $[s]$ has exactly one minimal event written $\min(s)$.
(2) Whenever $s_1 \sharp s_2$ in S, $\min(s_1) = \min(s_2)$.

This condition indeed ensures surjective pairing:

PROPOSITION 4.13. *Let $\sigma : \mathcal{S} \rightharpoonup !A^{\perp} \parallel !(B \parallel C)$ be a negative single-threaded $\sim$-strategy. Then, we have:*

$$\sigma \cong \langle \pi_1 \circledcirc \sigma, \pi_2 \circledcirc \sigma \rangle$$

PROOF. First of all, we define two subsets of S as follows:

$$\begin{array}{rcl} S_B & = & \{s \in S \mid \sigma(\min(s)) \in B\} \\ S_C & = & \{s \in S \mid \sigma(\min(s)) \in C\} \end{array}$$

(we abuse notations slightly with $\in B, \in C$).

By single-threadedness, S_B and S_C are disjoint and down-closed, with no immediate conflict spanning both components. Via projection, S_B and S_C can be seen as event structures and we have thus that $S \cong S_B \parallel S_C$. Moreover, it is direct to check that the restrictions of σ (along with a simple relabeling to $!B/!C$)

$$\sigma_B : S_B \rightharpoonup !A^{\perp} \parallel !B \qquad\qquad \sigma_C : S_C \rightharpoonup !A^{\perp} \parallel !C$$

are receptive and courteous, *i.e.* are strategies.

This decomposition also works at the level of symmetries. Any $\theta \in \tilde{S}$ preserves S_B and S_C. Indeed if $(s_B, s_C) \in \theta$, then $(\min(s_B), \min(s_C)) \in \theta$ as well: absurd, since one maps to $!B$ and the other to $!C$. It follows that $\theta = \theta_B \parallel \theta_C$ where θ_B and θ_C are bijections between configurations of S_B and S_C respectively. The set of restrictions to S_B (resp. S_C) of symmetries in $\tilde{S}$ yields a set of bijections between configurations of S_B (resp. S_C), which is easily checked to satisfy the conditions for an isomorphism family $\widetilde{S_B}$ (resp. $\widetilde{S_C}$). The labeling functions σ_B and σ_C preserve symmetry. Strong-receptivity and thinness follow directly from those for σ, so σ_B and σ_C are (negative) $\sim$-strategies.

By construction, σ_B and σ_C have disjoint codomain; so we can form their pairing $\langle\!\langle \sigma_B, \sigma_C \rangle\!\rangle$ without relabeling. Then, the aforementioned bijection $S = S_B \parallel S_C$ extends to a strong isomorphism of $\sim$-strategies:

$$\sigma \cong \langle\!\langle \sigma_B, \sigma_C \rangle\!\rangle$$

By Lemma 4.11, we have $\pi_1 \circledcirc \sigma \cong \sigma_B$ and $\pi_2 \circledcirc \sigma \cong \sigma_C$. But $\langle\!\langle \sigma_B, \sigma_C \rangle\!\rangle \cong \langle \sigma_B, \sigma_C \rangle$, and $\langle -, - \rangle$ preserves weak isomorphism, so we have surjective pairing for σ. □

This prompts us to look at the category of single-threaded, and negative $\sim$-strategies. For such an object to exist, we need first to check that single-threadedness is stable under composition. Since this property is independent from symmetry, we state the problem in the setting before symmetry:

PROPOSITION 4.14. *Let A, B, C be negative arenas, $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be negative single-threaded pre-strategies. Then, $\tau \circledcirc \sigma$ is single-threaded.*

PROOF. We first prove by induction on φ that for any secured bijection $\varphi : x_S \parallel (y_T)_\star \parallel x_C \simeq y_A \parallel (x_S)_\star \parallel y_T$ representing a configuration of $T \circledast S$, φ can partitioned into a finite union of disjoint φ_i (for $1 \leq i \leq n$) where each φ_i has one minimal event. Indeed, assume φ extends to φ' via (c, d) and φ' fails this condition. Necessarily, either c represents a nonnegative event s in S or d a nonnegative event in T. Assume the former. Then, the immediate *visible* predecessors of (c, d) in $\leq_{\varphi'}$ must be $(c_1, d_1), \ldots, (c_n, d_p)$ (using Lemma 2.45) where all the c_i are negative events s_i of S and $s_i \twoheadrightarrow s$.

By hypothesis, there are $1 \leq i, j \leq p$ and distinct $1 \leq k \neq l \leq n$ such that $(c_i, d_i) \in \varphi_k$ and $(c_j, d_j) \in \varphi_l$. But φ_k (resp. φ_l) must contain an event synchronized with $\min(s_i)$ (resp. $\min(s_j)$). Since σ is single-threaded and $s_i, s_j \in [s]$ we have $\min(s_i) = \min(s_j)$, which contradicts $\varphi_l \cap \varphi_k = \emptyset$.

Now, we go on to prove single-threadedness.

(1) Prime secured bijections have no non-trivial decomposition as above, therefore they have a unique minimal event. This is true in particular for the *visible* prime secured bijections. Condition *(1)* of single-threadedness follows from $T \circledcirc S$ being negative: its minimal event are always negative.

(2) Finally, assume there is a minimal conflict $\varphi \smile \psi$ in $T \circledcirc S$ between *visible* prime secured bijections. This means that there are prime secured bijections $\varphi' \subseteq [\varphi]_{T \circledast S}$, $\psi' \subseteq [\psi]_{T \circledast S}$, such that $\varphi' \smile \psi'$ in $T \circledast S$. Writing φ'' (resp. ψ'') for φ' (resp. ψ') without its top event, minimality of $\varphi' \smile \psi'$ means that $\varphi'' \cup \psi''$ is a valid secured bijection. Therefore, it decomposes:

$$\varphi'' \cup \psi'' = \biguplus_{1 \leq i \leq n} \varpi_i$$

With each ϖ_i a secured bijection having exactly one minimal event. If $n = 1$, we are done since as remarked the unique minimal event is necessarily visible. Otherwise, there are at least two ϖ_i, ϖ_j with distinct minimal events.

Then, using Lemma 2.45, $\varphi' \smile \psi'$ implies that their top elements are in minimal conflict in S or in T. Assume the former: the top element of φ corresponds to $s_\varphi \in S$ and that of ψ to $s_\psi \in S$ with $s_\varphi \smile_S s_\psi$. By receptivity and courtesy of σ, we have $\mathrm{pol}(s_{\varphi'}) = \mathrm{pol}(s_{\psi'}) = \star$. Since $n \geq 2$ and σ is single-threaded, courteous and receptive, there is a visible predecessor $s_1^- \in S$ of s_φ in φ and $s_2^- \in S$ of $s_\psi \in \varphi$ belonging to distinct ϖ_i and ϖ_j. Since $s_1 \leq s_\varphi$ and $s_2 \leq s_\psi$, we must

have $\min(s_1) = \min(s_2)$ by single-threadedness of σ. There must be an event in φ corresponding to this minimal event, but such an event would be in both ϖ_i and ϖ_j that were assumed disjoint. □

At this point we have everything to build a cartesian category. A **CHO-strategy** on an arena A is a negative, single-threaded, $\sim$-strategy on A.

PROPOSITION 4.15. *The following subcategory* $\mathrm{CHO}^{\cong}_{\circledcirc}$ *of* $\sim$-$\mathrm{tCG}_{\circledcirc}$ *is cartesian:*

- Objects: *negative arenas*,
- Morphisms from A to B: *CHO-strategies on* $!A^{\perp} \parallel !B$. *(up to weak isomorphism)*

This category will be written simply CHO in the rest since no variations are considered. If σ is a CHO-strategy (ie. a particular representative up to weak isomorphism) from A to B, we will write $\sigma : A \overset{\mathrm{CHO}}{\nrightarrow} B$. All operations on CHO (related to the cartesian structure) are actually defined on the representatives and then lifted to the isomorphism classes.

As in any cartesian category, given $\sigma_1 : A \overset{\mathrm{CHO}}{\nrightarrow} B$ and $\sigma_2 : C \overset{\mathrm{CHO}}{\nrightarrow} D$, we write $\sigma_1 \times \sigma_2 = \langle \sigma_1 \circledcirc \pi_1, \sigma_2 \circledcirc \pi_2 \rangle : A \parallel C \overset{\mathrm{CHO}}{\nrightarrow} B \parallel D$. This is not to be confused with the functorial action of $\parallel$ in $\sim$-$\mathrm{tCG}_{\circledcirc}$: $\sigma_1 \times \sigma_2$ plays on $!(A \parallel C)^{\perp} \parallel !(B \parallel D)$ whereas $\sigma_1 \parallel \sigma_2$ plays on $(!A \parallel !C)^{\perp} \parallel (!B \parallel !D)$. However, there is a canonical isomorphism of event structures with symmetry $m_{A,B} : !(A \parallel B) \cong !A \parallel !B$. This isomorphism can be used to relate the two actions:

LEMMA 4.16. *Let* $\sigma_1 : A \overset{\mathrm{CHO}}{\nrightarrow} C$ *and* $\sigma_2 : B \overset{\mathrm{CHO}}{\nrightarrow} D$. *We have a weak isomorphism:*

$$\sigma_1 \times \sigma_2 \cong \overline{m^{-1}_{C,D}} \circledcirc (\sigma_1 \parallel \sigma_2) \circledcirc \overline{m_{A,C}}$$

PROOF. Direct consequence of Lemma 3.53, since all mentioned compositions involve a lifted map. □

2.2. Cartesian-closure. We now move on to proving the category is cartesian-closed. The arrow of negative arenas A and B will be given by the arrow construction $A \Rightarrow B$ (as $A^{\perp} \parallel B$ need not be negative). To prove that this is actually the closure for $\parallel$, we use the fact that in our restricted space of strategies the game $!(A \Rightarrow B)$ and $!A^{\perp} \parallel !B$ support the same strategies and hence we can take advantage of the closure of $\sim$-$\mathrm{tCG}^{\cong}_{\circledcirc}$.

First, $!A^{\perp} \parallel !B$ is less causally constrained than $!(A \Rightarrow B)$, so we can build a map of event structure from the latter to the former:

LEMMA 4.17. *There is a strong-receptive, courteous map of essps:*

$$\chi_{A,B} : !(A \Rightarrow B) \to !A^{\perp} \parallel !B$$

which, additionally, preserves the copy index of negative events.

PROOF. For events $b \in B$ we use $\sharp b$ for the natural number associated to b by the countability of B. As in Section 1, we use $\langle - \rangle : \mathbb{N}^* \to \mathbb{N}$ for any injective function; the collision with the pairing operation should not generate any confusion.

We set:

$$\begin{array}{rcccl} \chi_{A,B} & : & !(A \Rightarrow B) & \to & !A^\perp \parallel !B \\ & & (\alpha : [(0,(b,a))] \to \mathbb{N}) & \mapsto & (0,\alpha') \\ & & (\beta : [(1,b)] \to \mathbb{N}) & \mapsto & (1,\beta') \end{array}$$

where:

$$\begin{array}{rcccll} \alpha' & : & [a] & \to & \mathbb{N} & \\ & & a' & \mapsto & \langle \sharp b, \alpha((1,b)), \alpha((0,(b,a'))) \rangle & (\text{if } a' \in \min(A)) \\ & & a' & \mapsto & \alpha((0,(b,a'))) & (\text{otherwise}) \end{array}$$

and:

$$\begin{array}{rcccl} \beta' & : & [b] & \to & \mathbb{N} \\ & & b' & \mapsto & \beta(1,b') \end{array}$$

With this definition $\chi_{A,B}$ preserves symmetry, is strong-receptive (it does not change the copy indices of negative events, since minimal events of $A^\perp$ are positive) and courteous (it only breaks immediate causal links from minimal events of B to minimal events of $A^\perp$, so from negative to positive). □

This allows us, from $\sigma : \mathcal{S} \rightharpoonup !C^\perp \parallel !(A \Rightarrow B)$, to define its relabeling:

$$\begin{array}{rcl} \Phi(\sigma) & : & \mathcal{S} \rightharpoonup !C^\perp \parallel (!A^\perp \parallel !B) \\ & = & (!C^\perp \parallel \chi_{A,B}) \circ \sigma \end{array}$$

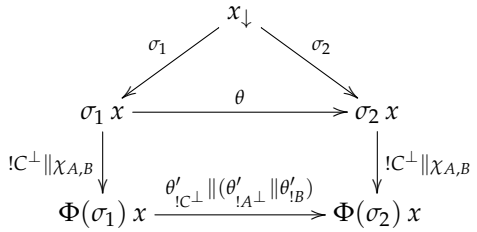
Before going on to the other direction, we note a further property of this relabeling.

LEMMA 4.18. *Let $\sigma : \mathcal{S} \rightharpoonup !C^\perp \parallel !(A \Rightarrow B)$ be a CHO-strategy. Take $s_1, s_2 \in S$ such that $\sigma\, s_1$ has the form $(1,\beta)$ (in the $A \Rightarrow B$ component) with* lbl $\beta = (1,b)$ *($b \in \min(B)$), and $\sigma\, s_2 = (1,\alpha)$ with* lbl $\alpha = (0,(b',a))$. *Then, $b = b'$ iff $s_1 = \min(s_2)$.*

PROOF. Straightforward consequence of single-threadedness. □

LEMMA 4.19. *Let $\sigma_1, \sigma_2 : \mathcal{S} \rightharpoonup !C^\perp \parallel !(A \Rightarrow B)$ be two CHO-strategies sharing the same internal ess. Then, $\sigma_1 \sim^+ \sigma_2$ iff $\Phi(\sigma_1) \sim^+ \Phi(\sigma_2)$.*

PROOF. *if.* Assume $\Phi(\sigma_1) \sim^+ \Phi(\sigma_2)$. Take $x \in \mathscr{C}(S)$, and form $\theta = \{(\sigma_1\, s, \sigma_2\, s) \mid s \in x_\downarrow\}$. We wish to prove that θ is a valid symmetry on $!C^\perp \parallel !(A \Rightarrow B)$. Firstly, we remark that the following diagram of bijections commutes.



where θ' is the bijection obtained from $\Phi(\sigma_1) \sim^+ \Phi(\sigma_2)$.

It follows that θ decomposes as $\theta_{!C^\perp} \parallel \theta_{!(A \Rightarrow B)}$ with $\theta_{!C^\perp} \in \widetilde{!C^\perp}$, and we are left to prove that $\theta_{!(A \Rightarrow B)} \in \widetilde{!(A \Rightarrow B)}$. By construction it is a bijection, so we need to prove that it preserves and reflects causality, that it preserves labels, and that it preserves indices of negative events – which is clear, as they are preserved throughout this diagram.

We prove that it preserves immediate causality. The only nontrivial case concerns immediate causal links not preserved by $\chi_{A,B}$, *i.e.* those of the form:

$$\sigma_1\, s_1 = (2, \{(2,b) \mapsto n\}) \rightarrowtriangle (2, \{(2,b) \mapsto n, (1,(b,a)) \mapsto p\}) = \sigma_1\, s_2$$

But then, by Lemma 4.18, we have $s_1 = \min(s_2)$. Since labels are preserved by $\theta'_{!A^\perp}$ and $\theta'_{!B}$, and using Lemma 4.18 again, we still have $\theta\,(\sigma_2\, s_1) \rightarrowtriangle \theta\,(\sigma_2\, s_2)$. The argument also applies to the θ^{-1}, which therefore is an order-isomorphism.

Preservation of labels also follows from Lemma 4.18. Finally, θ is a positive symmetry as all bijections involved preserve copy index of negative events.

only if. By preservation of symmetry for $\chi_{A,B}$, and the fact that it preserves the copy index of negative events. □

Relabeling from $!C^\perp \parr (!A^\perp \parr !B)$ to $!C^\perp \parr !(A \Rightarrow B)$ is slightly more subtle: indeed, we go from a game having one copy of A to one having as many as there are minimal moves in B. Thus, choosing the label for events formerly mapping to A requires us to choose a copy of A corresponding to some minimal event in B. Here condition *(1)* of single-threadedness is crucial: each move s mapped to A has a unique minimal dependency $\min(s)$, which must be mapped to a minimal event of B, and hence specifies the copy of A that s should be sent to. More formally, we prove the following.

LEMMA 4.20. *For any CHO-strategy $\sigma : \mathcal{S} \rightharpoonup !C \parr (!A^\perp \parr !B)$, there is $\sigma' : \mathcal{S} \rightharpoonup !C \parr !(A \Rightarrow B)$, unique up to positive symmetry, such that*

$$\sigma \sim^+ (!C \parr \chi_{A,B}) \circ \sigma'$$

PROOF. We define $\sigma' : \mathcal{S} \rightharpoonup !C \parr !(A \Rightarrow B)$. The domain of σ' is that of σ. For $s \in S_\downarrow$, then if $\sigma(s) = (0,\gamma)$ we set $\sigma'(s) = (0,\gamma)$.

If $\sigma(s) = (1,(1,\beta))$ with $\beta : [b] \to \mathbb{N}$, then we set $\sigma'(s) = (2,\beta')$ with

$$\begin{array}{rcccc} \beta' & : & [(1,b)] & \to & \mathbb{N} \\ & & (1,b') & \mapsto & \beta(b') \end{array}$$

If $\sigma(s) = (1,(0,\alpha))$ with $\alpha : [a] \to \mathbb{N}$, then by condition *(1)* of single-threadedness it has a unique minimal dependency $\min(s) \leq s$. By hypothesis, $\sigma\,(\min(s))$ has the form $(1,(1,\beta))$ with $\beta = \{b \mapsto n\}$. Therefore we set:

$$\begin{array}{rcccc} \alpha' & : & [(0,(b,a))] & \to & \mathbb{N} \\ & & (0,(b,a')) & \mapsto & \alpha(a') \\ & & (1,b) & \mapsto & n \end{array}$$

and we define $\sigma'(s) = (2,\alpha')$.

It is routine to check that this map is strong-receptive and courteous, and that its composition with $!C \parr \chi_{A,B}$ is positively symmetric to σ. It follows from Lemma 4.19 that it preserves symmetry, and that it is unique up to positive symmetry. □

From that, we deduce the following.

PROPOSITION 4.21. *There is a bijection Φ up to weak isomorphism, preserving and reflecting weak isomorphism, between:*

- *Negative, single-threaded $\sim$-strategies $\sigma : \mathcal{S} \rightharpoonup !C^\perp \parr !(A \Rightarrow B)$,*
- *Negative, single-threaded $\sim$-strategies $\sigma' : \mathcal{S} \rightharpoonup !C^\perp \parr (!A^\perp \parr !B)$.*

Moreover this bijection is compatible with pre-composition: for all $\tau : \mathcal{T} \multimap !D^\perp \parr !C$*:*

$$\Phi(\sigma) \circledcirc \tau \cong \Phi(\sigma \circledcirc \tau)$$

PROOF. On the one hand $\Phi(\sigma)$ is obtained as $(!C^\perp \parr \chi_{A,B}) \circ \sigma$, while $\Phi^{-1}(\sigma')$ is obtained by the unique factorization of Lemma 4.20. The bijection up to weak isomorphism follows from Lemma 4.20 as well.

We now prove stability under composition. By definition, we have $\Phi(\sigma) = (!C^\perp \parr \chi_{A,B}) \circ \sigma$. But by Lemma 3.53 this is the same (up to isomorphism) as $\overline{\chi_{A,B}} \circledcirc \sigma$, so the action of Φ can be obtained by post-composition via a lifted map. Stability under composition follows by associativity of composition. □

And finally, we deduce:

THEOREM 4.22. *The category* CHO *is cartesian closed.*

PROOF. We already know that it is cartesian. Throughout this proof, in the construction of the components of the cartesian closed structure, we ignore the associativity and unity isomorphisms from the compact closed structure of $\sim$-tCG$^{\cong}_{\circledcirc}$ – those can be easily and uniquely recovered from the context.

For any two arenas A, B, we first define the *evaluation* $\sim$-strategy (composition is in $\sim$-tCG$^{\cong}_{\circledcirc}$):

$$\begin{array}{rcl} \mathrm{ev}_{A,B} & : & A \parr (A \Rightarrow B) \overset{\mathrm{CHO}}{\nrightarrow} B \\ & = & (\epsilon_{!A} \parr !B) \circledcirc (!A \parr \Phi(\alpha_{!(A \Rightarrow B)})) \circledcirc \overline{m_{A,A \Rightarrow B}} \end{array}$$

where $\epsilon_{!A} \in \sim\text{-tCG}^{\cong}_{\circledcirc}((!A \parr !A^\perp)^\perp, 1)$ comes from $\sim$-tCG$^{\cong}_{\circledcirc}$ being compact-closed.

Likewise, for any $\sigma : A \parr C \overset{\mathrm{CHO}}{\nrightarrow} B$, we define its *curryfication* as:

$$\begin{array}{rcl} \Lambda(\sigma) & : & C \overset{\mathrm{CHO}}{\nrightarrow} (A \Rightarrow B) \\ & = & \Phi^{-1}(!A^\perp \parr (\sigma \circledcirc \overline{m_{A,C}}^{-1}) \circledcirc (\eta_{!A} \parr !C)) \end{array}$$

where $\eta_{!A} \in \sim\text{-tCG}^{\cong}_{\circledcirc}(1, (!A \parr !A^\perp))$ comes from $\sim$-tCG$_{\circledcirc}$ being compact-closed.

It is then a straightforward equational reasoning to prove the two equations [**LS88**], for $\sigma : A \times C \overset{\mathrm{CHO}}{\nrightarrow} B$,

$$\begin{array}{lrcl} (\beta) & \mathrm{ev}_{A,B} \circledcirc (A \times \Lambda(\sigma)) & \cong & \sigma \\ (\eta) & \Lambda(\mathrm{ev}_{A,B} \circledcirc (A \times \sigma)) & \cong & \sigma \end{array}$$

using mainly Proposition 4.21 and the laws of the compact closed structure of $\sim$-tCG, in combination with Lemma 4.16 to relate the cartesian structure of CHO$_{\circledcirc}$ and the monoidal structure of $\sim$-tCG – all the structural isomorphisms involved in the definition cancel each other. □

2.3. Recursion. To conclude the section, we prove that CHO supports the interpretation of a fixpoint combinator.

Usually in game semantics, the interpretation of the fixpoint combinator $\mathcal{Y}$ is obtained by showing that the category of games and strategies is enriched over a category of sufficiently complete partial orders. Here however it will not be the case: indeed, just as in AJM games [**AJM00**], our cartesian-closed category is a quotient (its morphisms being weak isomorphism classes). It is not clear how to build a complete ordering on weak isomorphism classes. However, this is not a big

issue: although weak isomorphism classes of $\sim$-strategies might not form a complete partial order, concrete $\sim$-strategies do. Therefore, when solving recursive strategy equations, we will make sure to work with concrete $\sim$-strategies rather than weak isomorphism classes.

Our first step will be to order concrete $\sim$-strategies.

DEFINITION 4.23. Let $\sigma : \mathcal{S} \rightharpoonup \mathcal{A}$, $\tau : \mathcal{T} \rightharpoonup \mathcal{A}$ be two $\sim$-strategies on a tcg $\mathcal{A}$. We write $\sigma \trianglelefteq \tau$ iff $S \subseteq T$, the inclusion map $\mathcal{S} \hookrightarrow \mathcal{T}$ is a map of essps, with all data in $\mathcal{S}$ coinciding with the restriction of that in $\mathcal{T}$, and such that for all $s \in S$, $\sigma\, s = \tau\, s$ (and both are equi-defined).

The $\sim$-strategies on $\mathcal{A}$ ordered by $\trianglelefteq$ form a directed complete partial order (dcpo). It is not *pointed* though – it does not have a least element. Indeed, a $\trianglelefteq$-minimal $\sim$-strategy must still satisfy receptivity, and hence comprise events for minimal negative events of $\mathcal{A}$. However, the *name* in $\mathcal{S}$ given to those is arbitrary, so there is one $\trianglelefteq$-minimal $\sim$-strategy on $\mathcal{A}$ for each renaming of the minimal negative events of $\mathcal{A}$. For each $\mathcal{A}$ we distinguish one $\trianglelefteq$-minimal $\sim$-strategy

$$\bot_{\mathcal{A}} : \min^{-}(\mathcal{A}) \to \mathcal{A}$$

that has as events the negative minimal events of $\mathcal{A}$ with induced symmetry, and as labeling function the identity. Not every $\sim$-strategy is above $\bot_{\mathcal{A}}$. However, for every $\sim$-strategy σ, we pick one $\sigma \cong \sigma^{\dagger}$ such that $\bot_{\mathcal{A}} \trianglelefteq \sigma^{\dagger}$ obtained by renaming the minimal negative events of σ. We write $\mathcal{D}_{\mathcal{A}}$ for the pointed dcpo of $\sim$-strategies above $\bot_{\mathcal{A}}$.

LEMMA 4.24. *For any tcg $\mathcal{A}$, $\mathcal{D}_{\mathcal{A}}$ is a pointed dcpo with $\bot_{\mathcal{A}}$ as minimal element.*

PROOF. If $\Gamma = \{\gamma : \mathcal{S}_{\gamma} \to \mathcal{A}\} \subseteq \mathcal{D}_{\mathcal{A}}$ is a directed subset of $\mathcal{D}_{\mathcal{A}}$, we form

$$\vee\Gamma = \cup\gamma : \bigcup_{\gamma \in \Gamma} S_{\gamma} \to \mathcal{A}$$

with all components defined as component-wise union.

This defines a $\sim$-strategy, which is the least upper bound of Γ. □

Additionally, we note that if all $\sim$-strategies in a directed set Γ are negative or single-threaded, so is $\vee\Gamma$. We now note that all the operations we defined on $\sim$-strategies in this section are continuous for $\trianglelefteq$.

LEMMA 4.25. *Composition, tensor, pairing, curryfication and the $(-)^{\dagger}$ operation defined above are continuous for $\trianglelefteq$.*

PROOF. Straightforward. □

From the above, we deduce the following.

COROLLARY 4.26. *For any arena A there is a fixpoint combinator $\mathcal{Y}_A : (A \Rightarrow A) \overset{\text{CHO}}{\rightharpoonup} A$,* i.e. *a single-threaded $\sim$-strategy such that:*

$$\mathcal{Y}_A \cong ev_{A,A} \circledcirc \langle \mathcal{Y}_A, \text{æ}_{!(A \Rightarrow A)} \rangle$$

PROOF. First, using the CCC structure of CHO, we define the sequence:

$$\sigma_0 = \bot \qquad \sigma_{n+1} \approxeq \text{ev} \circledcirc \langle \sigma_n, \text{æ}_{A \Rightarrow A} \rangle^{\dagger}.$$

Note that σ_n is (weakly isomorphic to) the interpretation of $\lambda f.\, f\,(\ldots\,(f\,\bot))$ given by the CCC structure of CHO. By induction, using Lemma 4.25 and $\bot = \sigma_0 \trianglelefteq \sigma_1$, it follows that $\sigma_i \trianglelefteq \sigma_{i+1}$. As a result, we let

$$\mathcal{Y} = \bigcup_{n\in\mathbb{N}} \sigma_n,$$

which satisfies the desired equation by standard reasoning. □

3. Adequate interpretations of ndPCF

In this section, we prove that CHO supports two interpretations of **ndPCF**. One interpretation is *sequential* and the other one is *parallel*: they only differ by the interpretation of the `if` construct. Both interpretations are shown to be adequate for both may and must convergences.

3.1. Semantics of terms and observational equivalence. We have seen in Section 1 the interpretation of types of our language. To complete this, we need to give the interpretation of terms. A term $\Gamma \vdash t : A$ will be interpreted as a CHO-strategy $[\![t]\!]$ on $![\![\Gamma]\!]^\bot \parr ![\![A]\!]$. It is crucial that the interpretation of a term is an actual strategy, and not an equivalence class, for the developments to come. We build two such interpretations $[\![t]\!]_{\mathtt{par}}$ and $[\![t]\!]_{\mathtt{seq}}$ that only differ by their interpretation of conditionals.

3.1.1. *The two interpretations of **ndPCF**.* The interpretation functions are built by induction on the syntax as usual:

(*λ-calculus*) Since CHO is a cartesian-closed category, we get automatically an interpretation of the simply-typed λ-calculus. For completeness, we recall here a sketch of the interpretation, see **[LS88]** for the full details:

$$[\![\Gamma, x : A \vdash x : A]\!] = \pi_2 : ([\![\Gamma]\!] \parr [\![A]\!]) \overset{\mathrm{CHO}}{\rightarrowtriangle} [\![A]\!]$$
$$\text{(well-defined since } [\![\Gamma, x : A]\!] = [\![\Gamma]\!] \parr [\![A]\!])$$
$$[\![\Gamma \vdash \lambda x.\, t : A \Rightarrow B]\!] = \Lambda([\![x : A, \Gamma \vdash t : B]\!]) : [\![\Gamma]\!] \overset{\mathrm{CHO}}{\rightarrowtriangle} [\![A \Rightarrow B]\!]$$
$$[\![\Gamma \vdash t\, u : B]\!] = \mathrm{ev} \circledcirc \langle [\![\Gamma \vdash t : A \Rightarrow B]\!], [\![\Gamma \vdash u : A]\!] \rangle : [\![\Gamma]\!] \to [\![B]\!]$$

(*Fixpoints*) The fixpoint operator is simply interpreted by that of CHO:

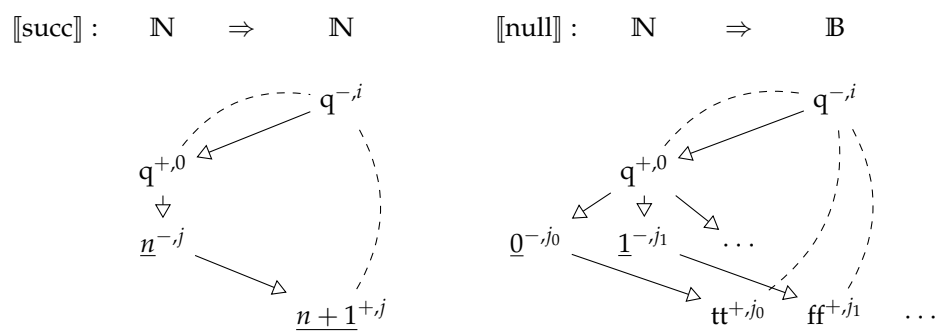
$$[\![\Gamma \vdash \mathcal{Y} : (A \Rightarrow A) \Rightarrow A]\!] = \Lambda(\mathcal{Y}_A) \circledcirc e_{[\![\Gamma]\!]}$$

(*Ground types*) The interpretation of values are as follows:

$$[\![\mathrm{tt}]\!] : \quad !\mathbb{B} \qquad [\![\mathrm{ff}]\!] : \quad !\mathbb{B} \qquad [\![\underline{n}]\!] : \quad !\mathbb{N}$$

$$\mathrm{q}^{-,i} \qquad \mathrm{q}^{-,i} \qquad \mathrm{q}^{-,i}$$
$$\downarrow \qquad \downarrow \qquad \downarrow$$
$$\mathrm{tt}^{+,0} \qquad \mathrm{ff}^{+,0} \qquad \underline{n}^{+,0}$$

The symmetry is inherited from that on $!\mathbb{B}$ and $!\mathbb{N}$. The strategy for `choice` has been given at Example 4.6. The integer operators succ and null are defined as follows (pred is defined similarly):



From those, we derive:

$$\llbracket \Gamma \vdash \mathrm{succ}\, M : \mathbb{N} \rrbracket = \mathrm{succ} \circledcirc \llbracket \Gamma \vdash M : \mathbb{N} \rrbracket$$
$$\llbracket \Gamma \vdash \mathrm{pred}\, M : \mathbb{N} \rrbracket = \mathrm{pred} \circledcirc \llbracket \Gamma \vdash M : \mathbb{N} \rrbracket$$
$$\llbracket \Gamma \vdash \mathrm{null}\, M : \mathbb{B} \rrbracket = \mathrm{null} \circledcirc \llbracket \Gamma \vdash M : \mathbb{N} \rrbracket.$$

Finally, conditionals are interpreted using two different strategies depicted in Figure 5, $\mathtt{if}_{\mathtt{par}}, \mathtt{if}_{\mathtt{seq}} : !\mathbb{B}^{\perp} \parr !\mathbb{B}^{\perp} \parr !\mathbb{B}^{\perp} \parr \mathbb{B}$:

$$\llbracket \mathtt{if}\, M\, N_1\, N_2 \rrbracket_{\mathtt{par}} = \mathtt{if}_{\mathtt{par}} \circledcirc \langle \llbracket M \rrbracket, \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle$$
$$\llbracket \mathtt{if}\, M\, N_1\, N_2 \rrbracket_{\mathtt{seq}} = \mathtt{if}_{\mathtt{seq}} \circledcirc \langle \llbracket M \rrbracket, \llbracket N_1 \rrbracket, \llbracket N_2 \rrbracket \rangle.$$

The strategy $\mathtt{if}_{\mathtt{par}}$ is *not* sequential: the three arguments are evaluated in parallel; and when the boolean evaluates to true and the first branch evaluates to a boolean, then `if` returns that boolean at top-level (and similarly for the other branch). However, the two implementations cannot be told apart by contexts arising from nondeterministic PCF (as a consequence of the adequacy results proved later in this section). In the rest of the section, we use the notation $\llbracket \cdot \rrbracket$ to mean *any of the two interpretations*: a statement holds for both interpretations.

3.1.2. *Testing equivalences.* We now define semantic counterparts to may and must convergences, and deduce the corresponding testing equivalences. A strategy $\sigma \in \mathrm{CHO}(\mathbb{X})$ **may converge** when it contains a positive move. Two strategies $\sigma, \tau \in \mathrm{CHO}(A)$ are **may-equivalent** ($\sigma \simeq_{\mathrm{may}} \tau$) when for all strategies $\alpha \in \mathrm{CHO}(A, \mathbb{B})$, $\alpha \circledcirc \sigma$ may converge if and only if $\alpha \circledcirc \tau$ may converge.

For must convergence, it is less clear how to generalize the syntactic notion. Consider the term $\Omega' = \mathcal{Y}(\lambda x.\, \mathtt{ifchoice}\, x\, x)$ that diverges while performing infinitely many nondeterministic choices. In particular both $\llbracket \Omega' \rrbracket_{\mathtt{par}}$ and $\llbracket \Omega' \rrbracket_{\mathtt{seq}}$ must not converge. However $\mathtt{if}\, \mathtt{tt}\, \mathtt{tt}\, \Omega'$ must converge, but its parallel interpretation is infinite since it runs Ω' (but does not wait on it): it even has an infinite configuration. To define must equivalence, we look at maximal configurations: they correspond to maximal reductions. Such maximal configurations must contain a positive move – otherwise, a maximal configuration without positive event is a witness of a reduction that cannot lead to a value.

Write $\mathscr{C}^{\infty}(E)$ for the set of finite or infinite configurations of an event structure E. A $\sigma \in \mathrm{CHO}(\mathbb{B})$ **must converge** if all its (possibly infinite) maximal configurations contain a positive move. Two strategies $\sigma, \tau \in \mathrm{CHO}(A)$ are **must-equivalent** when for all strategies $\alpha \in \mathrm{CHO}(A, \mathbb{B})$, $\alpha \circledcirc \sigma$ must converge if and only if $\alpha \circledcirc \tau$

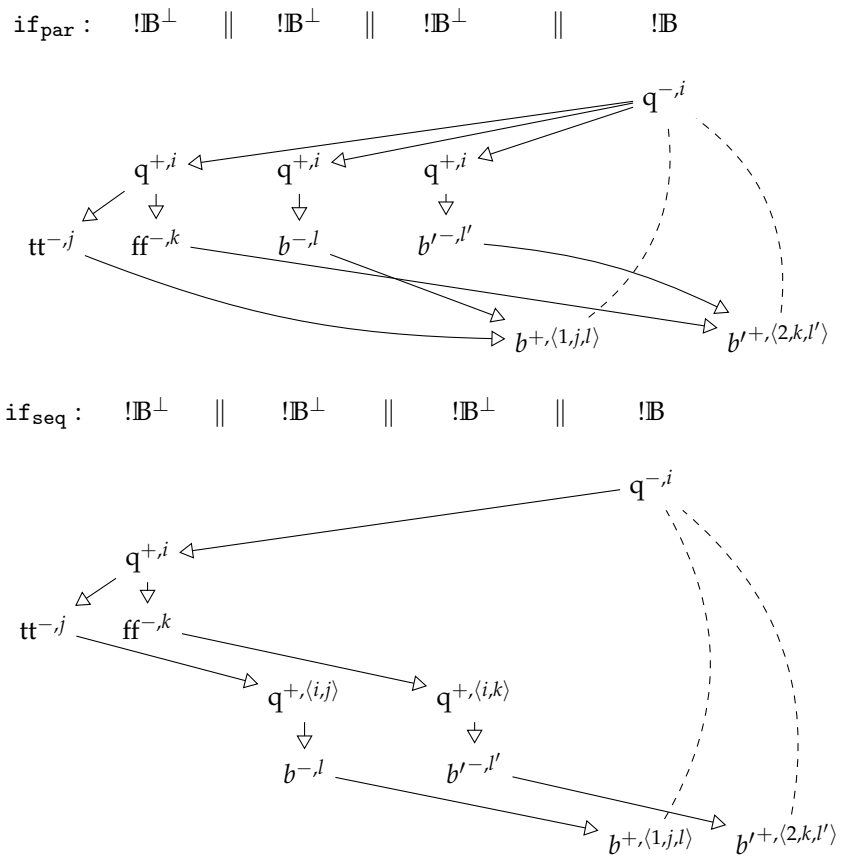


FIGURE 5. Two interpretations of `if`

must converge. Similarly, we write $\sigma \simeq_{\mathrm{m\&m}} \tau$ when σ and τ are both may and must equivalent.

It is key to look at infinite configurations: for instance the term

$$\mho = \mathcal{Y}(\lambda n.\, \mathtt{if}\ \mathtt{choice}\, 0\, (\mathrm{succ}\, n))$$

which returns a nondeterministic integer must not converge but all finite configurations of $[\![\mho]\!]_{\mathtt{seq}}$ can be extended with a positive move.

Remark that any syntactic context yields a semantic test: any $\mathcal{C}[\,]$ for type $\Gamma \vdash A$ gives a strategy $[\![\mathcal{C}]\!] = [\![\lambda h.\mathcal{C}[h\, x_1 \dots\, x_n]]\!] \in \mathrm{CHO}_{\circledcirc}([\![\Gamma]\!] \Rightarrow [\![A]\!], [\![\mathbb{B}]\!])$ where $\Gamma = x_1 : A_1, \dots, x_n : A_n$.

3.2. Relation between the operational and denotational semantics.

3.2.1. *The reduced part.* As an anticipation of Chapter 6, we take advantage of uniformity to cut down part of strategies that are redundant. Given a ground type $\mathbb{X}$, write $\mathbb{X}^+$ for the corresponding arena without the initial question. Any CHO-strategy $\sigma : \mathcal{S} \rightharpoonup\, !\mathbb{X}$ on $\mathbb{X}$ induces a $\sim$-strategy: $\mathfrak{r}(\sigma) : \mathfrak{r}(\mathcal{S}) \rightharpoonup\, !\mathbb{X}^+$ where $\mathfrak{r}(\mathcal{S})$ is

defined as ($\mathrm{q}_0 \in S$ is the initial question with copy index zero):

$$S \downarrow \{s \in S \mid s > \mathrm{q}_0\},$$

where $\mathfrak{r}(\sigma)$ is obtained by restriction. Symmetry on $\mathfrak{r}(S)$ is reduced to identities. Axioms of $\sim$-strategies are easily checked. The $\sim$-strategy $\mathfrak{r}(\sigma)$ is called the **reduced part** of σ. It characterizes σ:

LEMMA 4.27. *For a CHO-strategy $\sigma : \mathcal{S} \nrightarrow !\mathbb{X}$, $\sigma \approxeq \sigma'$ if and only $\mathfrak{r}(\sigma) \approxeq \mathfrak{r}(\sigma')$.*

PROOF. Consequence of uniformity. □

3.2.2. *The reduction tree.* From now on, unless mention of the contrary, we only consider closed terms of a ground type $\mathbb{X}$. The operational semantics on those terms also induces a $\sim$-pre-strategy on $!\mathbb{X}^+$ as follows.

A (non-necessarily finite) sequence $M_0 \to M_1 \to \ldots \to M_n$ is called a **reduction path**. We write $M_0 \xrightarrow{\pi}\!\!\!\!\rightarrow M_n$ to introduce a finite reduction path π from M_0 to M_n, and $M \xrightarrow{\pi}\!\!\!\!\rightarrow \ldots$ for an infinite reduction path starting at M. Reduction paths are naturally ordered by prefix ordering. As a result, finite reduction paths sharing the same starting point naturally organize themselves into a tree, regarded as an event structure $\mathfrak{t}(M)$ defined as follows:

- *Events*: non-empty *finite* reduction paths of the form $\eta(M) \xrightarrow{\pi}\!\!\!\!\rightarrow M'$.
- *Causality*: prefix ordering of paths,
- *Conflict*: two paths are in conflict if they are not comparable.

where $\eta(M) = M[\texttt{if choice tt ff/choice}]$. The labelling function $\mathfrak{t}(M) \nrightarrow !\mathbb{X}^+$ is only defined on finite paths π that end on a value v. In that case, the corresponding move of the game is v with copy index $\sharp(\pi)$ where $\sharp(\cdot)$ is a hashing function, from reduction paths to natural numbers. The resulting labelling function gives a pre-$\sim$-strategy $\mathfrak{t}(M)$ on $!\mathbb{X}^+$ (secrecy holds because of the η-expansion of `choice` performed above).

3.2.3. *Finite approximations.* To handle terms with fixpoints, the notion of *finite approximation* comes in handy. Given a term M, we write M_n for M where occurrences of $\mathcal{Y}$ were substituted by $\mathcal{Y}_n$ (interpreted in the model as $[\![\lambda f.\, f^n \perp]\!]$), along with the reduction rules:

$$\mathcal{Y}_{n+1}\, M \to M\,(\mathcal{Y}_n\, M) \quad \text{and} \quad \mathcal{Y}_0\, M \to \perp.$$

Because composition is continuous and we have that $[\![\mathcal{Y}_n]\!]_{\mathtt{seq}} \approxeq \sigma_n$ (defined in 2.3), we have that $[\![M]\!]_{\mathtt{seq}} \approxeq \bigcup_{n\in\mathbb{N}} [\![M_n]\!]^{\dagger}_{\mathtt{seq}}$ (likewise $[\![M]\!]_{\mathtt{seq}} \approxeq \bigcup_{n\in\mathbb{N}} [\![M_n]\!]^{\dagger}_{\mathtt{seq}}$).

This makes sense since $[\![M_n]\!]_{\mathtt{seq}}$ is a particular representative and not an equivalence class. As a result, we have:

$$\mathfrak{r}[\![M]\!]_{\mathtt{seq}} = \bigcup_{n\in\mathbb{N}} \mathfrak{r}[\![M_n]\!]_{\mathtt{seq}}.$$

Finally, any path $M_n \xrightarrow{\pi}\!\!\!\!\rightarrow N$ for $N \neq \perp$ yields a path $M_{n+1} \xrightarrow{\pi'}\!\!\!\!\rightarrow N'$ by rewriting the subterms of the shape $\mathcal{Y}_k$ into $\mathcal{Y}_{k+1}$. As a result $\mathfrak{t}(M_n)$ can embed as a down-closed subtree of $\mathfrak{t}(M_{n+1})$. In the following, this embedding will be silent and we will assume that $\mathfrak{t}(M_n)$ *is* a subtree of $\mathfrak{t}(M_{n+1})$. As a result:

$$\mathfrak{t}(M) \approxeq \bigcup_{n\in\mathbb{N}} \mathfrak{t}(M_n)$$

3.2.4. *Must convergence and the reduction tree.* By looking at the reduction tree, we can prove that must convergence is finitary in the following sense:

LEMMA 4.28. *For a term $\vdash M : \mathbb{B}$, the following are equivalent:*

(1) M must converge,
(2) $\mathfrak{t}(M)$ is finite,
(3) $\mathfrak{t}(M)$ is finite and all leaves are positive event labelled by values,
(4) there exists M_n such that M_n must converge.

PROOF. *(1) $\Rightarrow$ (2)* Since M must converge and $\mathfrak{t}(M)$ is v finitely-branching, $\mathfrak{t}(M)$ is finite by König's lemma.

(3) $\Rightarrow$ (1) Obvious by the definition of reduction trees.

(2) $\Rightarrow$ (3) Corollary of Lemma 4.1.

(3) $\Rightarrow$ (4) Since $\mathfrak{t}(M) \cong \bigcup_{n\in\mathbb{N}} \mathfrak{t}(M_n)$, and $\mathfrak{t}(M)$ is finite, this union is reached in finite time. Hence there exists M_n such that $\mathfrak{t}(M_n) \cong \mathfrak{t}(M)$, and M_n must converge.

(4) $\Rightarrow$ (2) If M_n must converge, then its reduction tree must be finite and all leaves must be positive events. As a result $\mathfrak{t}(M_n) \cong \mathfrak{t}(M)$. □

3.3. Adequacy. We now prove a strong link between the operational semantics and the denotational semantics: for the sequential interpretation, they compute essentially the same tree (Theorem 4.29). Such strong results tying the denotational semantics and the operational semantics are, to our knowledge, new and rely heavily on the essential events. This link allows us to deduce adequacy for may and must for both interpretations as a corollary.

3.3.1. *Sequential interpretation.* To prove adequacy of the sequential interpretation, we relate $\mathfrak{t}(M)$ to $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$. In general $\mathfrak{t}(M)$ has more events than $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ which only retains the events where a nondeterministic choice was made. Generalizing the notation of Chapter 2 to $\sim$-strategies, we write $\mathscr{E}(\mathfrak{t}(M))$ for

$$\mathfrak{t}(M) \downarrow \{s \mid s \text{ is involved in a minimal conflict or is visible}\}.$$

It is another $\sim$-strategy on $!\mathbb{X}^+$. The interesting property of our model is that this strategy, obtained from the operational semantics, exactly coincide the one obtained denotationally:

THEOREM 4.29. *For any term $\vdash M : \mathbb{X}$, there exists a weak isomorphism*

$$\varphi : \mathfrak{r}[\![M]\!]_{\mathtt{seq}} \cong \mathscr{E}(\mathfrak{t}(M))$$

Theorem 4.29 is proved using realizability. Say that a term M realizes a ground type $\mathbb{X}$ if it satisfies the condition of the theorem, and M realizes $A \Rightarrow B$ if for any realizer N of A, $M\,N$ realizes B.

This definition satisfies the usual β-expansion lemma:

LEMMA 4.30 (β-expansion). *Let $x : A \vdash M : B$ be a term. If $M[N/x]$ realizes B for some realizer N of A, then $(\lambda x.\, M)\, N$ realizes B.*

PROOF. By induction on B. The only interesting case is the base case. In that case, we have

$$\mathfrak{r}[\![(\lambda x.\, M')\, N]\!]_{\mathtt{seq}} \cong \mathfrak{r}[\![M'[N/x]]\!]_{\mathtt{seq}} \cong \mathscr{E}(\mathfrak{r}(M'[N/x])) \cong \mathscr{E}(\mathfrak{t}((\lambda x.\, M)\, N)),$$

where the second isomorphism is the induction hypothesis, and the third comes from $(\lambda x.\, M')\, N$ only reducing to $M'[N/x]$. □

LEMMA 4.31. *For every (open) term $x_1 : A_1, \dots, x_n : A_n \vdash M : B$ without fixpoint, and terms $\vdash N_i : A_i$ such that N_i realizes A_i, then $M[N_i/x_i]$ realizes B.*

PROOF. We proceed by induction on M.

Constants: straightforward, both event structures are reduced to a single event, labelled with the corresponding value.

Choice: We have that $\eta(M) \to \mathtt{if}\,\mathtt{tt}\,\mathtt{tt}\,\mathtt{ff} \to \mathtt{tt}$ and $\eta(M) \to^2 \mathtt{ff}$ which means that $\mathscr{E}(\mathfrak{t}(M))$ and $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ are isomorphic.

Application and variable: straightforward.

λ-abstraction: Consequence of Lemma 4.30.

Ground type operators: For instance assume $M = \mathrm{succ}\, M_0$. Write M' for $M[N_i/x_i]$ and M'_0 for $M_0[N_i/x_i]$. It is a direct calculation to see that $\mathfrak{r}[\![M']\!]_{\mathtt{seq}}$ is isomorphic to $\mathfrak{r}[\![M'_0]\!]_{\mathtt{seq}}$ as an event structure, with the labelling function to $!\mathbb{X}^+$, as follows: $\mathrm{lbl}(\mathfrak{r}[\![M']\!]_{\mathtt{seq}}) = \mathrm{succ} \circ \mathrm{lbl}(\mathfrak{r}[\![M'_0]\!]_{\mathtt{seq}})$ where we wrote $\mathrm{lbl}(\cdot)$ to denote explicitly the labelling function of the corresponding strategies.

Similarly, $\mathfrak{t}(M')$ is similar to $\mathfrak{t}(M)$ but each leaf labelled by n is replaced by a tree of the shape $\star \twoheadrightarrow 1 + n$ since it requires one more step to compute. This disappears when applying $\mathscr{E}(\cdot)$ hence $\mathfrak{t}[\![M']\!]_{\mathtt{seq}} \cong \mathscr{E}(\mathfrak{t}(M'))$ by induction hypothesis.

Conditionals: the conditional is implemented by a sequential strategy so the same reasoning as in the previous case applies. □

We can now complete the proof of Theorem 4.29, using finite approximations:

PROOF. (Of theorem 4.29) Let $\vdash M : \mathbb{X}$ be a closed term of ground type. By Lemma 4.31, M_n realizes $\mathbb{X}$, and there is a family of isomorphisms

$$\varphi_n : \mathfrak{r}[\![M_n]\!]_{\mathtt{seq}} \cong \mathfrak{t}(\mathscr{E}(M_n)).$$

Unfortunately, the φ_n might not agree with each other. For instance, both event structures corresponding to $\mathtt{if}\,\mathtt{choice}\,\mathtt{tt}\,\mathtt{tt}$ have non-trivial isomorphisms.

First, since the $\mathfrak{t}(\mathscr{E}(M_n))$ are all trees, it follows that the $\mathfrak{r}[\![M_n]\!]_{\mathtt{seq}}$ are also trees, and so is $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$. Moreover, they are finitely branching (since we can only make a binary choice at a time), this means that for any $e \in \mathfrak{r}[\![M]\!]_{\mathtt{seq}}$, there are only a finite number of elements that it can be mapped to by the φ_n.

Now, we build $\varphi : \mathfrak{r}[\![M]\!]_{\mathtt{seq}} \cong \mathscr{E}(\mathfrak{t}(M))$ by induction on $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ by maintaining the invariant that for each e, there exists infinitely many $n \in \mathbb{N}$ such that $\forall e' \leq e, \varphi e' = \varphi_n e'$. Assume we have built the image of the predecessors of e. Consider the set $\{\varphi_n(e) \mid n \in \mathbb{N}, \forall e' \leq e, \varphi_n(e') = \varphi(e')\}$. It is non-empty by induction and finite by the remark above, so there must be an element of the set that corresponds to infinitely many n. Choose $\varphi(e)$ to be such an element – by construction it satisfies the invariant. □

The result of Theorem 4.29 implies adequacy of the sequential interpretation.

THEOREM 4.32. *Let M be a closed term of ground type $\mathbb{X}$. We have the following:*

- *M may converge if and only if $[\![M]\!]_{\mathtt{seq}}$ may converge*
- *M must converge if and only if $[\![M]\!]_{\mathtt{seq}}$ must converge.*

PROOF. *May convergence.* Assume that $M \to^* v$ with v being a value. By construction, $\mathfrak{t}(M)$ has a move labelled with v which means by Theorem 4.29 that $[\![M]\!]_{\mathtt{seq}}$ contains a positive move. Conversely, if $[\![M]\!]_{\mathtt{seq}}$ has a positive move, then so does $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ and $\mathfrak{t}(M)$ which means that there exists v such that $M \to^* v$.

Must convergence. If M must converge, then by Lemma 4.28 its reduction tree is finite with positive leaves. As a result, by Theorem 4.29, $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ has the same shape and $[\![M]\!]_{\mathtt{seq}}$ must converge.

Conversely, by Theorem 4.29, $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ is a tree, with positive leaves. Any infinite reduction path π of M would yield a configuration x of $\mathfrak{r}[\![M]\!]_{\mathtt{seq}}$ that can be extended to x' containing a positive move s. Since π is infinite, x cannot contain a positive move, and $s \notin x$. Consider a path $M \xrightarrow{\pi'\!\!\!\!\!\!\rightarrow} v$ with $\varphi(s) = \pi'$. Write π_0 for the longest common prefix to π and π'. By construction π_0 can extend by a prefix π_1 of π and a prefix π_1' of π'. That $\pi_1 \frown \pi_1'$ in $\mathfrak{t}(M)$ and $\{\varphi^{-1}(\pi_1), \varphi^{-1}(\pi_1')\} \subseteq x'$, are contradictory with φ being an isomorphism. □

3.3.2. *Concurrent interpretation.* We deduce adequacy of the concurrent interpretation from the sequential one by means of a logical relation. Define a relation $\sigma \sim_A \sigma'$ where $\sigma, \sigma' \in \mathrm{CHO}(A)$ by induction on types as follows:

$$\begin{array}{rcl} \sigma \sim_{\mathbb{X}} \sigma' & \text{iff} & (\sigma \Downarrow_{\mathrm{must}} \Leftrightarrow \sigma' \Downarrow_{\mathrm{must}}) \wedge \sigma \simeq_{\mathrm{may}} \sigma' \\ \sigma \sim_{A \Rightarrow B} \sigma' & \text{iff} & \forall \tau \sim_A \tau', \sigma \circledcirc \tau \sim_B \sigma' \circledcirc \tau' \end{array}$$

Then, the fundamental lemma for this logical relation implies adequacy:

LEMMA 4.33. *Let $x_1 : A_1, \dots, x_n : A_n \vdash M : A$ be a term of **ndPCF** and $\sigma_i \sim_{A_i} \sigma_i'$ be related strategies. Then*

$$[\![M]\!]_{\mathtt{par}} \circledcirc \langle \sigma_1, \dots, \sigma_n \rangle \sim_A [\![M]\!]_{\mathtt{seq}} \circledcirc \langle \sigma_1', \dots, \sigma_n' \rangle$$

As a result, for a closed term $\vdash M : \mathbb{X}$, if $[\![M]\!]_{\mathtt{par}}$ must converge (resp. may converge), then M must converge (resp. may converge) by adequacy of the sequential interpretation.

PROOF. Most cases are dealt with in standard way. The only interesting case is `if`: assume that $M = \mathtt{if}\ N\ N_1\ N_2$, and without loss of generality that M is closed.

By inspection of the sequential and concurrent interpretations of `if`, we have:

$$\begin{array}{rcl} [\![\mathtt{if}\ N\ N_1\ N_2]\!]_{\mathtt{seq}} \Downarrow_{\mathrm{must}} & \text{iff} & [\![N]\!]_{\mathtt{seq}} \Downarrow_{\mathrm{must}} \wedge (\mathtt{tt} \in [\![N]\!]_{\mathtt{seq}} \Rightarrow [\![N_1]\!]_{\mathtt{seq}} \Downarrow_{\mathrm{must}}) \\ & & \wedge (\mathtt{ff} \in [\![N]\!]_{\mathtt{seq}} \Rightarrow [\![N_2]\!]_{\mathtt{seq}} \Downarrow_{\mathrm{must}}) \\ [\![\mathtt{if}\ N\ N_1\ N_2]\!]_{\mathtt{par}} \Downarrow_{\mathrm{must}} & \text{iff} & [\![N]\!]_{\mathtt{par}} \Downarrow_{\mathrm{must}} \wedge (\mathtt{tt} \in [\![N]\!]_{\mathtt{par}} \Rightarrow [\![N_1]\!]_{\mathtt{par}} \Downarrow_{\mathrm{must}}) \\ & & \wedge (\mathtt{ff} \in [\![N]\!]_{\mathtt{par}} \Rightarrow [\![N_2]\!]_{\mathtt{par}} \Downarrow_{\mathrm{must}}) \end{array}$$

from which the result follows by induction hypothesis. □

From these adequacy results, we deduce that observational equivalence in the semantics entails observational equivalence in the model.

COROLLARY 4.34. *The interpretations $[\![\cdot]\!]_{\mathtt{seq}}$ and $[\![\cdot]\!]_{\mathtt{par}}$ are both sound for may, must, and may&must equivalence.*

*Formally, for $\Gamma \vdash M, M' : A$ two terms of **ndPCF** such that $[\![M]\!]_{\mathtt{seq}}$ and $[\![M]\!]'_{\mathtt{seq}}$ are may-equivalent (resp. must-equivalent, may&must-equivalent), then M and M' are may-equivalent (resp. must-equivalent, may&must-equivalent), and similarly for $[\![\cdot]\!]_{\mathtt{par}}$.*

PROOF. The proofs for the three equivalences follow the same pattern. Assume for instance $[\![M]\!]_{\mathtt{seq}}$ and $[\![M]\!]'_{\mathtt{seq}}$ are may-equivalent. Let $\mathcal{C}[]$ be a context for type $\Gamma \vdash A$ such that $\mathcal{C}[M]$ may converge. Then $[\![\mathcal{C}[]]\!]_{\mathtt{seq}} \circledcirc [\![M]\!]_{\mathtt{seq}}$ may converge which implies that $[\![\mathcal{C}[]]\!]_{\mathtt{seq}} \circledcirc [\![M']\!]_{\mathtt{seq}}$ may also converge. If follows, by adequacy, that $\mathcal{C}[M']$ may also converge. □

Because of the expressive power of strategies, the converse is not true: two terms could be indistinguishable by **ndPCF** but their interpretation could be distinguished by some strategy. The next part is concerned with cutting down the space of strategies to ensure this does not happen, and isolate a sub-model of $\mathrm{CHO}_{\circledcirc}$ which is intensionally fully-abstract, *ie.* where the converse holds. This is done by generalizing the traditional notion of innocence and well-bracketing of play-based game semantics to our partial-order based setting.

Part 2

Innocence

In this second part, we generalize the notions of well-bracketing and innocence traditional in HO game semantics [**HO00**], in order to understand what properties strategies coming from a language without control operators (leading to the notion of *well-bracketing*) or state (leading to *innocence*) satisfy. We finally prove that the interpretations defined in Chapter 4 of **ndPCF** inside the class of well-bracketed and innocent strategies are intensionally fully abstract.

Plan of the part.

Chapter 5. This chapter introduces our conditions of well-bracketing and innocence, and proves that they are stable under composition so that we get subcartesian closed categories of CHO consisting in the innocent, and well-bracketed strategies. Moreover, in this chapter we show a very important property of *visible* strategies (a property weaker than innocence): their interaction is deadlock-free. This means that composition of visible strategies is relational.

Chapter 6. This chapter proves that our interpretations of **ndPCF** given in Chapter 4 are intensionally fully abstract (for may testing). In the process, key properties of innocent and well-bracketed properties are investigated. In particular, we show that innocent strategies support a reduced form which generalizes the P-view tree of strategies in HO games. This induces a notion of *finite* strategy. We also show that innocent and well-bracketed strategies on a higher-order type can be decomposed into smaller strategies of higher-order type and a strategy of first-order type. This allows us to reduce finite definability to finite definability at first-order types.

CHAPTER 5

Concurrent innocence and well-bracketing

In this chapter, we introduce conditions on strategies of CHO to restrict their discriminating power to that of the interpretation of **ndPCF**. Instead of defining one condition that will exactly capture the expressive power of **ndPCF**, we decompose it into several orthogonal conditions that capture bigger languages. In the sequential world, such a decomposition allows to understand orthogonality of computational effects. In particular, the condition of well-bracketing corresponds to the absence of control operators [**Lai99, Lai97**], visibility to the absence of higher-order state [**AHM98**] and innocence to the absence of ground state [**AM99a**]. Consequently, the expressive power of PCF is captured by the innocent, deterministic well-bracketed strategies.

Related work. The problem of concurrent well-bracketing was first addressed and solved by Ghica and Murawski in [**GM07**], where they obtain a full abstraction result for a concurrent language with shared memory, but without control operators. Our notion is simply a reunderstanding of theirs to our causal setting.

To our knowledge, there are no prior solutions to the problem of innocence in a nondeterministic and concurrent world. At the time this work was conducted, even the problem of nondeterministic sequential innocence was still an open problem, closed independently by [**TO15**] since then.

The first to extend innocence outside a sequential world are Melliès and Mimram [**MM07**] in the setting of concurrent and deterministic (non-alternating) strategies on asynchronous games. This innocence is expressed via switching conditions given by the structure of the type (in their case, a formula of linear logic). On the contrary, our notion of innocence is intrinsic in the spirit of traditional HO games. We would like to point out that the true concurrent setting of asynchronous games allows Melliès and Mimram to design a notion of innocence ensuring that composition of innocent strategies is deadlock-free. This is similar to what happens here, where visibility ensures deadlock-free interactions (Theorem 5.35).

The work of Hirschowitz *et al.* [**Hir14, EHS15**] also features a notion of innocence defined as a sheaf condition (since their strategies are defined as presheaves). This idea was recast in the λ-calculus by Ong and Tsukada giving the notion of innocence mentioned above. We believe that our notion of innocence can be formulated this way (since it is possible to regard our strategies as presheaves), although technical details have not been carried out.

Outline of the chapter. Our conditions do not make use of symmetry, so we define them in the setting without symmetry. They do make use of negativity, and single-threadedness, so we first start in Section 0 by introducing the subcategory of negative and single-threaded strategies of $\mathrm{CG}^{\cong}_{\circledcirc}$. Then, Section 1 generalizes the well-bracketing condition of [**GM07**] to our partial-order setting.

Section 2 introduces the problem of concurrent innocence and of its formulation in a partial-order setting. It also introduces *visibility* which is key for innocence to be stable under composition and *locality* restricting the shape of conflict.

Section 3 proves innocence and locality to be stable under composition and builds a category of innocent strategies, $\mathrm{CHO}_{\mathtt{inn}}$.

Contributions of this chapter. The results presented in this chapter are joint work with Pierre Clairambault. The conditions for the deterministic case are published in **[CCW15]** where they are proved to correspond to PCF. This chapter introduces the new extension of visibility, *locality* that generalizes visibility to nondeterminism, as well as a new definition of well-bracketing that is independent from visibility. These conditions of innocence and locality can be understood as the restriction between which parts of the program can communicate, whereas well-bracketing restricts which moves can be played (answering several times the same question, etc.)

0. Negative and single-threaded strategies

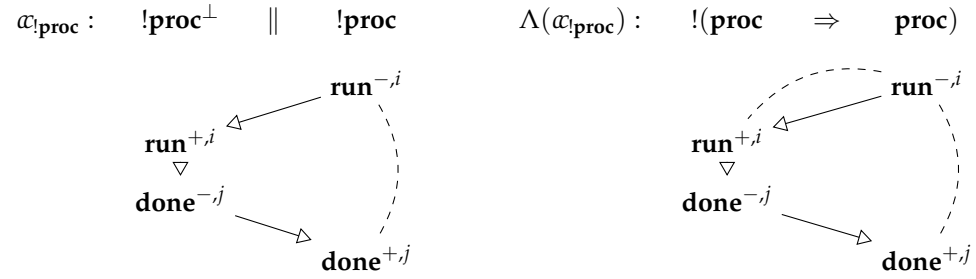
We now introduce the formal setting where the development of this chapter takes place. Because symmetry is not of concern when defining innocence and well-bracketing, we only consider for games negative arenas, and negative, single-threaded, essential strategies on them. They organize themselves naturally in a subcategory of $\mathrm{CG}^{\cong}_{\circledcirc}$:

PROPOSITION 5.1. *The following defines a subcategory* $\mathrm{nCG}^{\cong}_{\circledcirc}$ *of* $\mathrm{CG}^{\cong}_{\circledcirc}$
Objects: *negative arenas,*
Morphisms: *negative, single-threaded essential strategies on* $A^{\perp} \parallel B$.

Concrete CHO-strategies $\sigma : \mathcal{S} \rightharpoonup\, !A^{\perp} \parallel\, !B$ are erased to concrete strategies $\sigma : S \to\, !A^{\perp} \parallel\, !B$ in $\mathrm{nCG}^{\cong}_{\circledcirc}(!A, !B)$ in a composition-preserving way. This allows us to lift any condition stable under composition in $\mathrm{nCG}^{\cong}_{\circledcirc}$ to CHO.

Justifiers. A key notion throughout this chapter will be that of *justifiers*. Mentioned in passing in Chapters 3 and 4, they will be useful to formulate notions of innocence and well-bracketing suited to our setting. Indeed, justifiers are key to HO-based game semantics.

However, in the formulation of the previous chapters there is a slight hiccup. Justifiers are defined on strategies (as a partial map from S to the game), but depend on the target game. For instance, consider on the left $\mathfrak{a}_{!\mathbf{proc}} : !\mathbf{proc}^{\perp} \parallel\, !\mathbf{proc}$, and its curryfication playing on $!(\mathbf{proc} \Rightarrow \mathbf{proc})$:



We see here that, on the diagram on the left $\mathbf{run}^{+,j}$ has no justifier (since its image in the game is minimal), but on the right it has a justifier. Intuitively, if $\sigma \in \mathrm{CHO}(A)$, only minimal events do not have a justifier, but if $\sigma \in \mathrm{CHO}(A, B)$ this is not the case anymore. This is problematic when defining conditions on strategies that depend on the justifier structure. To solve this problem, we introduce an extended notion of justifiers, taking advantage of single-threadedness to define a justifier for those positive moves that are minimal in the game:

DEFINITION 5.2 (Extended justifier). Let $\sigma : S \rightharpoonup A$ be a partial map of event structures to an arena such that S is negative and single-threaded. For $s \in S_{\downarrow}$ non-minimal, we define its **extended justifier** $\mathrm{just}_{\mathrm{e}}(s)$ as follows:

- If σs is not minimal, then $\mathrm{just}_{\mathrm{e}}(s)$ is defined as the unique event $s' < s$ with $\sigma s' \rightarrowtriangle \sigma s$, exactly as before: $\mathrm{just}(s) = \mathrm{just}_{\mathrm{e}}(s')$,
- If σs is minimal: $\mathrm{just}_{\mathrm{e}}(s)$ is the unique minimal move of $[s]$ (well-defined by single-threadedness).

In the following, we will say that s **is justified by** s' when $\mathrm{just}_{\mathrm{e}}(s) = s'$. Note that, $\mathrm{just}_{\mathrm{e}}(e) < e$ for non-minimal e, and if σ is courteous, the extended justifier of a negative move is always its predecessor in S (if it exists).

LEMMA 5.3. *Let $\sigma : S \rightharpoonup A^{\perp} \parallel B$ and $\tau : T \rightharpoonup A^{\perp} \parallel B$ be courteous partial maps of event structures such that S and T are negative and single-threaded. Let $f : S \to T$ be a map of event structures such that $\{(\tau(fs), \sigma s) \mid s \in x\}$ defines an order-isomorphism $\tau(fx) \cong \sigma x$ for every $x \in \mathscr{C}(S)$. Then, for all non-minimal $s \in S$*

$$f(\mathrm{just}_{e}(s)) = \mathrm{just}_{e}(f(s))$$

The general statement of the lemma allows for f to be part of a weak isomorphism when put in the context of CHO.

PROOF. Let $s \in S_{\downarrow}$ non-minimal. First, notice that σs is non-minimal if and only if $\tau(fs)$ is.

If σs is non-minimal, then $\sigma(\mathrm{just}_{\mathrm{e}}(s))$ is the predecessor of σs in $[\sigma s]$. As a result, since $\tau(f(\mathrm{just}_{\mathrm{e}}(s))$ is the predecessor of $\tau(fs)$ in $\tau(f[s]) \cong \sigma[s]$, $f(\mathrm{just}_{\mathrm{e}}(s)) = \mathrm{just}_{\mathrm{e}}(fs))$ by definition of the justifier.

Otherwise, if σs is minimal, then so is $\tau(fs)$. Let t_0 be the minimal event of $[fs]$, and s_0 its unique pre-image in $[s]$ (that exists $[fs] \subseteq f[s]$ as f is a map of event structures). Since σ is a courteous, and σs_0 is negative and minimal, so must be s_0. Therefore $s_0 = \mathrm{just}_{\mathrm{e}}(s)$, and $f(\mathrm{just}_{\mathrm{e}}(s)) = t_0 = \mathrm{just}_{\mathrm{e}}(fs)$. □

Note that if σ, τ are strategies in $\mathrm{nCG}^{\cong}_{\circledcirc}$, the interaction $\tau \circledast \sigma$ is courteous negative and single-threaded, so extended justifiers apply to $\tau \circledast \sigma$. In particular, the projections preserve (extended) justifiers, by the previous lemma.

Grounded causal chains. Another technical tool that is crucial to define the conditions is that of *grounded causal chain*:

DEFINITION 5.4. A **grounded causal chain** (**gcc**) of an event structure S is a non-empty finite set $\varrho = \{\varrho_0, \ldots, \varrho_n\}$ with $\varrho_0 \in \min(S)$ and $\varrho_i \rightarrowtriangle_S \varrho_{i+1}$ for $i < n$.

Gccs represent sequential sub-part of the strategy, or *threads*. The set of gccs of an event structure S is written $\mathrm{gcc}(S)$. Gccs can be viewed as sets (as the definition) or as sequences since $\leq_S$ restricts to a linear ordering on them. Note that gccs of

strategies in $\mathrm{nCG}^{\cong}_{\circledcirc}$ are always alternating by courtesy and alternation of arenas. The length of a gcc ϱ will be denoted by $|\varrho|$, its last element by ϱ_ω and segments by $\varrho_{i\leq\_\leq k}$, $\varrho_{\leq i}$ or $\varrho_{\geq i}$.

We now move on to the definition of conditions.

1. Well-bracketing

In this section, we introduce *well-bracketing*, a condition restricting the call/return disciplines of strategies to match that of a programming language without control operators. Note that our definition differs from **[CCW15]**, which is specialized to a deterministic and innocent setting. We show in the next chapter, that up to observational equivalence, we can recover the conditions of **[CCW15]** in the presence of innocence (Proposition 6.15).

Questions and answers. The usual device in game semantics to formulate well-bracketing is that of *questions* and *answers*. Questions correspond to function or variable calls (Opponent or Player) and answers to function returns or variable values (again Opponent or Player). Each move of the arena is either a question or an answer, which is formalized as a labelling on arenas.

DEFINITION 5.5. An **arena with questions and answers** (or, in the following, **Q/A-arena**) is an arena A along with labelling map $A \to \{\mathscr{Q}, \mathscr{A}\}$ such that:

(1) initial moves are questions,
(2) answers are maximal.

In the rest of the thesis, we only consider Q/A arenas that we simply call arenas – replacing the previous notion. Constructions on arenas (arrow, product, expansion) trivially extend to Q/A-arenas. If $\sigma : S \rightharpoonup A \in \mathrm{nCG}^{\cong}_{\circledcirc}$ is a strategy on a Q/A-arena, the action of σ naturally induces a Q/A-labelling on $S_{\downarrow}$.

Terminology on questions/answer. If $S_{\downarrow}$ has a Q/A-labelling, we say that an answer $a \in S$ **answers** a question $\mathrm{q} \in S$ when $\sigma\mathrm{q} \twoheadrightarrow \sigma a$ (or, equivalently $\mathrm{just}(a) = \mathrm{q}$). A consistent set X is **complete** if every question in X has at least one answer in X, and **affine** if every question in X has at most one answer in X. Its **pending question** (if it exists) is the greatest unanswered question in X. To introduce new events that have a specific labelling we will use the notation "let $\mathrm{q}^{\mathscr{Q}}$" or "let $a^{\mathscr{A}}$".

1.1. Intuitions from Idealized Parallel Algol (IPA).

In **[GM07]**, a notion of well-bracketing is introduced that captures exactly the expressive power of Idealized Parallel Algol, a concurrent language with ground state. Our goal in this section is to illustrate and to justify by examples the challenges of concurrent well-bracketing that are solved in **[GM07]** and to deduce a definition of well-bracketing adapted to our setting.

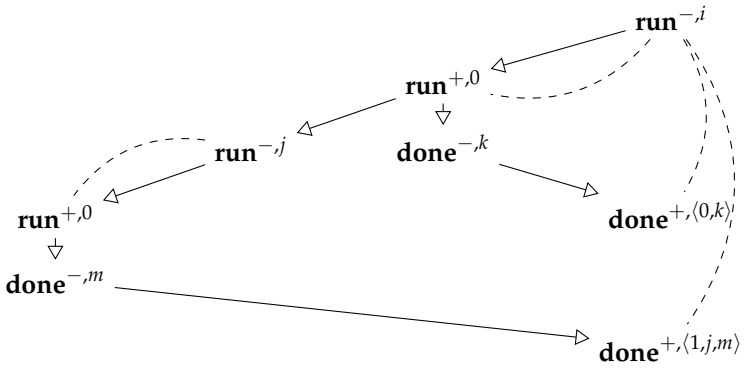
1.1.1. *Syntax of IPA.* IPA can be defined as the following extension of **ndPCF**:

$$
\begin{array}{rll}
A, B ::= & \cdots \mid \mathbf{proc} \mid \mathbb{N} \mid \mathbb{N}\,\mathbf{ref} & \\
M, N ::= & \cdots \mid \mathrm{new}\ r\ \mathrm{in}\ M & \text{(Reference declaration)} \\
& \mid\ !M \mid M := N \mid \mathtt{incr}\, M & \text{(Reference manipulation)} \\
& \mid M; N \mid \mathtt{skip} & \text{(Sequential composition and skip)} \\
& \mid M \parallel N & \text{(Parallel composition)}
\end{array}
$$

The extension adds references on natural numbers and parallel composition. We use wait (b) as a shorthand for $\mathcal{Y}(\lambda x.\, \texttt{if}\ b\ \texttt{skip}\ x)$ for the thread actively waiting for b to become true. IPA naturally has a type **proc** of commands interpreted inside CHO by the arena $\mathbf{run}^- \twoheadrightarrow \mathbf{done}^+$ (**run** being a question, and **done** an answer). CHO supports an interpretation of Idealized Algol [**CCW14**], although we only use it for illustrative purposes, to show that the patterns considered in this chapter can happen in real programming languages and are not simply quirks of the model.

1.1.2. *Well-bracketing in a sequential context.* In sequential HO game semantics, well-bracketing states that when a strategy plays an answer, its justifier must be the latest unanswered question. This typically rules out control operators, for instance, the following strategy for `call/cc`:

$$\texttt{call/cc}: \quad !(((\mathbf{proc} \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc})$$

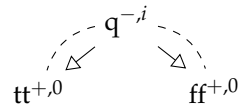


On this diagram, we see the last $\mathbf{done}^+$ being played despite $\mathbf{run}_j^-$ not being answered. This strategy can be used to discriminate terms of PCF by exploring branches of computation that might not terminate. For instance the terms $\lambda x.x; \bot$ and $\lambda x.\bot$ are distinguished by the context $\texttt{call/cc}\,(\lambda k.\,[]\,(k\,\texttt{skip}))$. However, those terms are indistinguishable by contexts of IPA.

This can be expressed in our setting by asking that the gccs of a strategy should be well-bracketed: an answer in the gcc should point to the latest unanswered question of the gcc. This condition is part of well-bracketing in [**CCW15**]. However, this condition is not well-behaved in general – an event can be present in a gcc but its justifier might not –, and not stable under composition without further conditions, eg. innocence. However, this is satisfied by terms of PCF as we will see later (Lemma 6.14). It is, however, not enough to cut down the expressive power to that of terms of IPA.

1.1.3. *Forking strategies.* Since there are no conflicts in the arena $\mathbb{B}$, the following diagram defines a valid strategy on $!\mathbb{B}$:

$$\texttt{fork}: \quad !\mathbb{B}$$



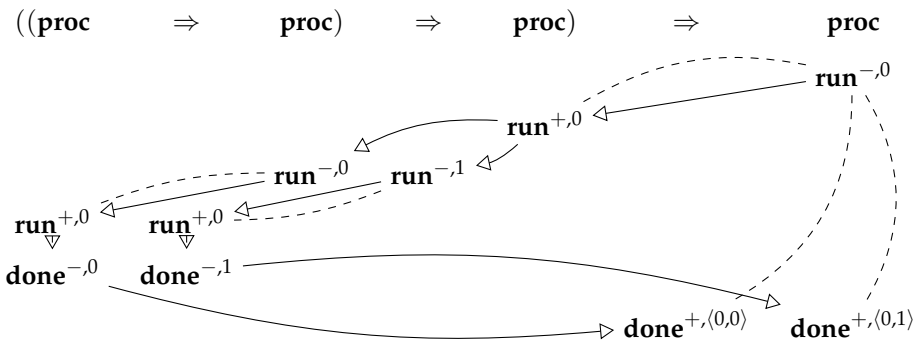


FIGURE 1. A configuration of `call/cc`

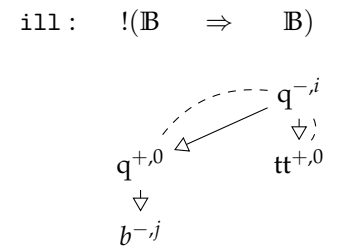
The gccs of this strategy are well-bracketed, but it answers twice *concurrently* to the toplevel. The consequence is that it will duplicate its execution context: one copy will see tt and the other ff. This strategy is a *a concurrent control operator*, as it is possible to define `call/cc` in an extension of IPA with `fork`. As a consequence, it can be used to distinguish observational equivalent terms of IPA.

Note that, receptivity implies that Opponent is allowed to answer multiple times. Say that a strategy σ is *fork-free* when for all configuration x of σ, if every positive question of x has at most one answer in x, then x is affine. This condition was also part of the well-bracketing condition of **[CCW15]**.

Note that in IPA, `fork` is also definable from `call/cc` by simply letting: `fork` $=$ `call/cc` $(\lambda k.\, k\,\mathrm{tt} \parallel k\,\mathrm{ff})$. This code makes two concurrent calls to its evaluation context (via `call/cc`), one call is given tt and the other ff. As a result, it must be that the strategy for `call/cc` must not be fork-free as well, even though it seems that `call/cc` never answers several times at toplevel! In fact, it does, but hidden behind our compact representation of strategies.

Figure 1 depicts a configuration of the strategy for `call/cc`, where Opponent never answered twice, but the initial question is answered twice by Player. In this configuration, Opponent called twice its continuation, resulting in two answers at toplevel. As a result, `call/cc` is not fork-free. Actually, this intuition will be key to show that *innocent* and *well-bracketed* strategies (to be defined later) have well-bracketed gccs (Lemma 6.14).

1.1.4. *Well-answered questions.* Answering twice the same question is not the only way to wrongly answer a question. For instance, the strategy `ill`



has a behaviour which is not IPA-definable. It answers the initial question, and in parallel interrogates its argument. As a result, the I(P)A-term:

$$M = \lambda f^{\mathbf{proc}\Rightarrow\mathbf{proc}}.\,\mathrm{new}\ r := 0\ \mathrm{in}\\ \qquad \mathtt{if}\ (f\,(r := 1; \bot))\\ \qquad\quad (!r = 1)\\ \qquad\quad \bot$$

that never converges on any IPA-definable function **proc** $\Rightarrow$ **proc**, converges when fed an argument behaving as the strategy `ill`. The problem here is that the initial question can be answered before the questions it justifies are answered. Such behaviours are forbidden by [**GM07**], and deemed not *well-answered*:

DEFINITION 5.6. Let $\sigma : S \rightharpoonup A$ be a map of event structures to an arena where S is negative and well-threaded. A question $\mathrm{q} \in S$ is **well-answered** in a configuration $x \in \mathscr{C}(S)$ when for all answer $a \in x$ to q and distinct $m \in x$ justified by q (ie. $\mathrm{just}_{\mathrm{e}}(m) = \mathrm{q}$), then m is a question answered in x.

As a result, a well-answered question in x is always answered at most once.

We can now define well-bracketing in our setting. As before, we cannot force Opponent to answer well all Player questions, so we should only force Player to answer well when Opponent does:

DEFINITION 5.7. A strategy $\sigma : S \rightharpoonup A \in \mathrm{nCG}^{\cong}_{\circledcirc}$ is **well-bracketed** when for all $x \in \mathscr{C}(S)$ such that all Player questions of x are well-answered in x, then all Opponent questions are well-answered in x.

Unlike the notion of well-bracketing introduced in [**CCW15**], this condition is stable under composition with no further assumptions on strategies, as we will see. It is clearly stable under parallel composition, and contains all the strategies $\sigma : A \to A \in \mathrm{nCG}^{\cong}_{\circledcirc}(A)$ where σ acts the identity on events (necessary to build the open interaction).

1.1.5. *Well-bracketing and causalities.* Well-bracketing puts some restrictions on the causal structure of S, as witnessed by the following lemma:

LEMMA 5.8. *Let $\sigma \in \mathrm{nCG}^{\cong}_{\circledcirc}(A, B)$ be a well-bracketed strategy. If $x \in \mathscr{C}(\sigma)$ is a complete and affine configuration, there are no negative answers that are maximal in x.*

PROOF. Assume there exists a complete and affine configuration x of S that contains a maximal negative answer s. It answers a question q_1^+, justified by a question q_0^-. Moreover, this question has an answer a^+ in x.

In $y := x \setminus \{s\} \in \mathscr{C}(S)$, q_0 is not well-answered since it is answered but q_1 is not (x is affine). However, all positive questions are well-answered since we only removed an answer to a positive question. This contradicts well-bracketing. □

In particular, this means that a well-bracketed strategy is not allowed to run a computation and discard the result concurrently to answering the initial question, ruling out the example presented above.

1.2. The category of well-bracketed strategies. In this section, we show that well-bracketed strategies form a subcategory of $\mathrm{nCG}^{\cong}_{\circledcirc}$. We first check copycat.

LEMMA 5.9. *The copycat strategy $\mathfrak{a}_A$ is well-bracketed for any arena A. As a consequence lifted strategies are well-bracketed.*

PROOF. For $(i, a) \in \mathfrak{C}_A$ let us write $\overline{(i,a)} = (1-i, a)$ for the corresponding event on the other component.

Let $x \parallel y \in \mathscr{C}(\mathfrak{C}_A)$ such that every positive question is well-answered and let q be a negative question in y (for instance) with an answer $a \in y$ to q and a move $m \in y$ justified by q. We know that $\mathrm{q} \rightarrowtriangle \overline{\mathrm{q}} \rightarrowtriangle \overline{m} \rightarrowtriangle m$ and similarly $\mathrm{q} \rightarrowtriangle \overline{\mathrm{q}} \rightarrowtriangle \overline{a} \rightarrowtriangle a$. In particular, $\overline{\mathrm{q}}$ is positive and thus well-answered: $\overline{m}$ is a question which is answered by $a_0 < \overline{a}$. It follows that m is a question, answered by $\overline{a_0}$ as desired. □

1.2.1. *Composition of well-bracketed strategies.* We now look at composition of well-bracketed strategies. Consider $\sigma : S \multimap A^\perp \parallel B$ and $\tau : T \multimap B^\perp \parallel C$, well-bracketed strategies of $\mathrm{nCG}^{\cong}_{\circledcirc}$. We first prove their interaction is well-behaved:

LEMMA 5.10. *Let z be a configuration of $T \circledast S$ such that questions mapped to a positive event of $A^\perp \parallel C$ are well-answered in z. Then any question is well-answered in z.*

PROOF. By induction on z, we show that for all pairs $m, a^{\mathscr{A}}$ in z, sharing the same justifier $\mathrm{q}^{\mathscr{Q}}$, that m is an answered question in z. If a and m are negative in $A^\perp \parallel C$, since q is positive, it must be well-answered by hypothesis.

Otherwise, since a and m share the same justifier, they must be either both σ-actions or τ-actions. Assume w.l.o.g that they are σ-actions (ie. project to positive moves in $S \parallel C^\perp$).

In $\Pi_1 z$, $\Pi_1 a$ and $\Pi_1 m$ share the same justifier. Assume that $\Pi_1 m$ is not answered in $\Pi_1 z$ (which is equivalent to m not being answered in z). By well-bracketing of $\sigma \parallel C$ applied to the configuration

$$x_0 = [\{\Pi_1 m, \Pi_1 a\}] \subseteq \Pi_1[\{m, a\}],$$

there exists a positive question $\Pi_1 \mathrm{q}' < \Pi_1 \mathrm{q}$ along with $(\Pi_1 a'^{\mathscr{A}}, \Pi_1 m')$ in x_0 such that a' answers q' and m', justified by q', is an answer or an unanswered question in x_0. Because they have different polarities, we have $a \neq a'$ and $m \neq m'$, hence the pair (a', m') lives in a configuration strictly smaller than z, and by induction hypothesis m' must be answered in $\Pi_1 x$: a contradiction. □

LEMMA 5.11. *The strategy $\tau \circledcirc \sigma$ is well-bracketed.*

PROOF. Take $z \in \mathscr{C}(T \circledcirc S)$ such that all positive questions of z are well-answered. As a result, the witness $[z] \in \mathscr{C}(T \circledcirc S)$ satisfies the conditions of Lemma 5.10. All questions in $[z]$ are thus all well-answered, as desired. □

1.2.2. *The category* $\mathrm{CHO}_{\mathtt{wb}}$. From the previous section, there is a subcategory of well-bracketed strategies that is monoidally-closed. But we are more interested in subcategories of CHO. Given a strategy $\sigma \in \mathrm{CHO}(A, B)$, we say it is **well-bracketed** when its erasure $\sigma \in \mathrm{nCG}^{\cong}_{\circledcirc}(!A, !B)$ is.

PROPOSITION 5.12. *Negative Q/A arenas and well-bracketed strategies form a cartesian closed subcategory* $\mathrm{CHO}_{\mathtt{wb}}$ *of* CHO.

PROOF. As seen above, copycat and all lifted maps are well-bracketed. Composition of well-bracketed strategies is well-bracketed by Lemma 5.11. By definition of extended justifiers, well-bracketing is invariant under currying. □

1.3. Observational equivalence in $\mathrm{CHO}_{\mathtt{wb}}$. It will be crucial in the next chapter to understand which parts of a well-bracketed strategy are observable by well-bracketed contexts (See Proposition 6.15). To do so, we now show that, up to (may and must) observational equivalence in $\mathrm{CHO}_{\mathtt{wb}}$, we can cut down parts of well-bracketed strategies that cannot be explored by such contexts.

1.3.1. *Complete part of a strategy.* Let us fix $\sigma : \mathcal{S} \rightharpoonup !A^{\perp} \parallel !B \in \mathrm{CHO}_{\mathtt{wb}}(A, B)$. Call a positive event $s \in S$ **observable** when it belongs to a complete and affine configuration. A negative event is **observable** when its justifier is observable or it is minimal. Observable visible events are enough to capture the behaviour up to may equivalence of σ. However, to capture the behaviour up to must, we also need to retain some essential events. An essential event $s \in S$ is **observable** when its (negative or essential) predecessors are observable.

LEMMA 5.13. *Observable events are closed under symmetry.*

PROOF. Follows from symmetries preserving causality and justification. □

Write S_{cmpl} for the set of observable events of S. By the previous lemma, we can consider $\mathcal{S}_{\mathrm{cmpl}} = \mathcal{S} \downarrow S_{\mathrm{cmpl}}$ and we get a map $\sigma_{\mathrm{cmpl}} : \mathcal{S}_{\mathrm{cmpl}} \rightharpoonup !A^{\perp} \parallel !B$.

By definition of observable events, S_{cmpl} is downward closed in S. In particular, we have that $\mathscr{C}(S_{\mathrm{cmpl}}) \subseteq \mathscr{C}(S)$ and all complete configurations of S are configurations of S_{cmpl}. As a result, we do get a well-bracketed strategy:

LEMMA 5.14. *The map σ_{cmpl} is a well-bracketed strategy.*

PROOF. Routine check. □

1.3.2. *Relation between σ and σ_{cmpl}.* We now show that σ and σ_{cmpl} are observationally equivalent in $\mathrm{CHO}_{\mathtt{wb}}$.

LEMMA 5.15. *Let $\sigma \in \mathrm{CHO}_{\mathtt{wb}}(A, B)$ and $\tau \in \mathrm{CHO}_{\mathtt{wb}}(B, C)$ be well-bracketed strategies. For $x \in \mathscr{C}(T \circledcirc S)$ a complete and affine configuration, $[x]$ is also complete and affine, and consequently, the configurations $\Pi_1[x]$ and $\Pi_2[x]$ are also complete and affine.*

PROOF. First, since x is complete and affine, then all questions of x are well-answered. By Lemma 5.10, so are all questions in $[x]$, hence $[x]$ is affine. To conclude, we show that any $\mathrm{q} \in [x]$ invisible question is answered in $[x]$. We proceed by induction on $[(\tau \circledast \sigma)\mathrm{q}] \in !B$.

If $(\tau \circledast \sigma)\mathrm{q}$ is not minimal in $!B$, then by induction hypothesis, its justifier is answered in $!B$. Otherwise, its justifier is an initial (visible) question which is answered since x in complete. In both cases, its justifier is well-answered (by the remark above) and answered in x, so q must be answered in x. □

THEOREM 5.16. *For $\sigma \in \mathrm{CHO}_{\mathtt{wb}}(A, B)$, the strategies σ and σ_{cmpl} are both may and must observationally equivalent for well-bracketed contexts, or more precisely for all well-bracketed $\alpha \in \mathrm{CHO}(A \Rightarrow B, \mathbb{B})$:*

(1) $\alpha \circledcirc \sigma$ may converge iff $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$ may,
(2) $\alpha \circledcirc \sigma$ must converge iff $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$ must.

PROOF. *(1)* Since σ_{cmpl} is obtained by removing events from σ, it easy to see that if $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$ may converge then so may $\alpha \circledcirc \sigma$.

Conversely, assume there exists a prime configuration $[p^+] \in \mathscr{C}(\alpha \circledcirc \sigma)$ with a top positive move. Since $\alpha \circledcirc \sigma$ plays on $\mathbb{B}$, it is easy to see that $[p^+]$ is complete and

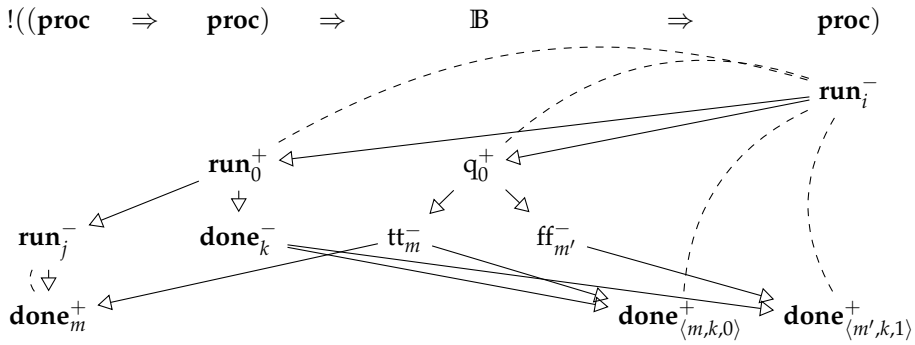
affine. Hence by Lemma 5.15, closing down yields a complete and affine witness $x \in \mathscr{C}(\alpha \circledast \sigma)$ whose first projection $\Pi_1 x \in \mathscr{C}(S)$ is also complete and affine. Since $\Pi_1 p \in \Pi_1 x$, p is observable and $p \in \alpha \circledcirc S_{\mathrm{cmpl}}$ and $\alpha \circledcirc S_{\mathrm{cmpl}}$ may converge.

(2) Now assume that $\alpha \circledcirc \sigma$ must converge. Let $x \in \mathscr{C}^\infty(\alpha \circledcirc \sigma_{\mathrm{cmpl}})$ be a maximal (possibly infinite) configuration. Since $\mathscr{C}^\infty(S_{\mathrm{cmpl}}) \subseteq \mathscr{C}^\infty(S)$, we have that $x \in \mathscr{C}^\infty(\alpha \circledcirc \sigma)$ hence x extends in $\mathscr{C}^\infty(\alpha \circledcirc \sigma)$ to a maximal x' where all questions are answered. This means that x' is complete (and must be affine because $\alpha \circledcirc \sigma$ is well-bracketed), so that s is observable. Hence, since $x \cup [s] \in \mathscr{C}^\infty(\alpha \circledcirc \sigma_{\mathrm{cmpl}})$ and x is maximal in $\mathscr{C}^\infty(\alpha \circledcirc \sigma_{\mathrm{cmpl}})$, it must be that $s \in x$ as desired.

Finally, assume that $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$ must converge, and let x be a maximal (possibly infinite) configuration of $\alpha \circledcirc \sigma$. Write $x' = \{p \in x \mid \Pi_1[p] \subseteq S_{\mathrm{cmpl}}\} \in \mathscr{C}^\infty(\alpha \circledcirc \sigma_{\mathrm{cmpl}})$. By assumption, x' can extend to $x'' \in \mathscr{C}^\infty(\alpha \circledcirc \sigma_{\mathrm{cmpl}})$ with a positive s.

A minimal event in $x'' \setminus x$ must be positive by definition of observable events. From there, by secrecy it follows that $x \cup x''$ is a valid (possibly infinite) configuration of $\alpha \circledcirc \sigma$ and by maximality, $x'' \subseteq x$ which implies that $s \in x$ as desired. □

So σ and σ_{cmpl} are observationally equivalent for the induced may and must observational equivalence in $\mathrm{CHO}_{\mathtt{wb}}$. However, this does not remove all the behaviours unreachable by well-bracketed contexts. For instance, consider the following well-bracketed strategy:



If Opponent is well-bracketed and answers ff, then we know that $\mathbf{run}_j^-$ will never be answered in a configuration where positive questions are well-answered. As a result, the configuration shown above cannot be reached in an interaction with a well-bracketed context, and removing it (for instance by adding a conflict between $\mathbf{done}^+_{\langle m',k,1\rangle}$ and $\mathbf{done}^+_m$) results in an observationally equivalent strategy.

Such examples show it is hard to exactly capture the set of complete configurations as an event structure built from S. However, although not precise enough, this construction will be useful in Chapter 6 to deduce interesting consequences of innocence and well-bracketing up to observational equivalence.

2. Towards a definition of concurrent innocence

In this section, we now investigate potential definitions of innocence in this concurrent and nondeterministic setting. We introduce our innocence condition in $\mathrm{nCG}^{\cong}_{\circledcirc}$ and show its lifting to CHO gives a sub-CCC of innocent strategies $\mathrm{CHO}_{\mathtt{inn}}$. In the previous section, we had a language to illustrate well-bracketed

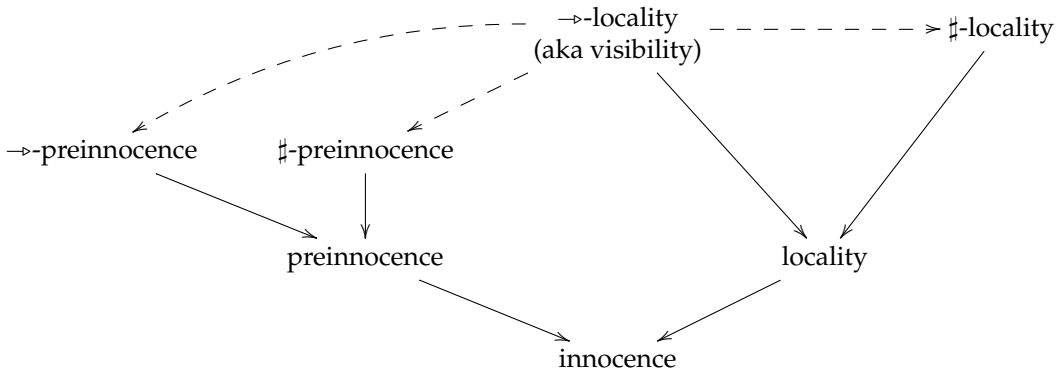
behaviours: IPA. However, to our knowledge, there is no concurrent and nondeterministic language to which innocent strategies should naturally correspond to (by means of a full-abstraction result). In particular, **ndPCF** is not suitable as it is *also* well-bracketed. However, we will use IPA to explain *informally* some of our counterexamples coming up in the rest of the section, and show that the causal patterns considered do appear already in IPA. The causal and conflict patterns can be encoded using references. For instance, for causal links it suffices to write on one end, and to wait for the writing to be performed on the other end. Hence, most causal patterns in $\mathrm{CHO}_{\mathtt{wb}}$ can be implemented this way. We would like to stress that the strategies depicted are **not** the interpretation of the terms provided, but simply that the term exhibits a causal pattern that is related to that of the strategy, that both the strategies *and* the terms should be considered informal evidence.

The purpose of innocence is to restrict that causal structure to match that of functional programs (without state). Note that, in comparison, well-bracketing was about restricting which moves can be played (eg. Player cannot play two answers to the same question, or answer a question before another).

In general we use the term interference (between moves, branches, etc.) to mean "causality or conflict" at an informal level. In the following, we try to look two kinds of interferences:

- interferences between syntactic branches (conditionals, arguments to the same calls) leading to our notion of **preinnocence**,
- interferences between concurrent subcomputations, leading to the notion of **locality**.

Each of these conditions is further broken down in a condition about causality ($\rightarrowtriangle$), and one about conflict ($\sharp$). The following diagram sums up our breakdown of innocence in a concurrent and non-deterministic setting:



An arrow $C \dashrightarrow C'$ means that C' is only stable under composition assuming C. When applicable, a condition is the conjunction of its predecessors for $\rightarrow$.

2.1. Interferences between external branches.

2.1.1. *Non-innocent behaviours.* Using state, one can create interferences between two Opponent branches: for instance two branches of the same conditional or two arguments of the same function call, or even two Opponent calls to the same variable. Such interferences can be used for instance to observe the number of times a function uses its argument. Figure 2 depicts two strategies exhibiting

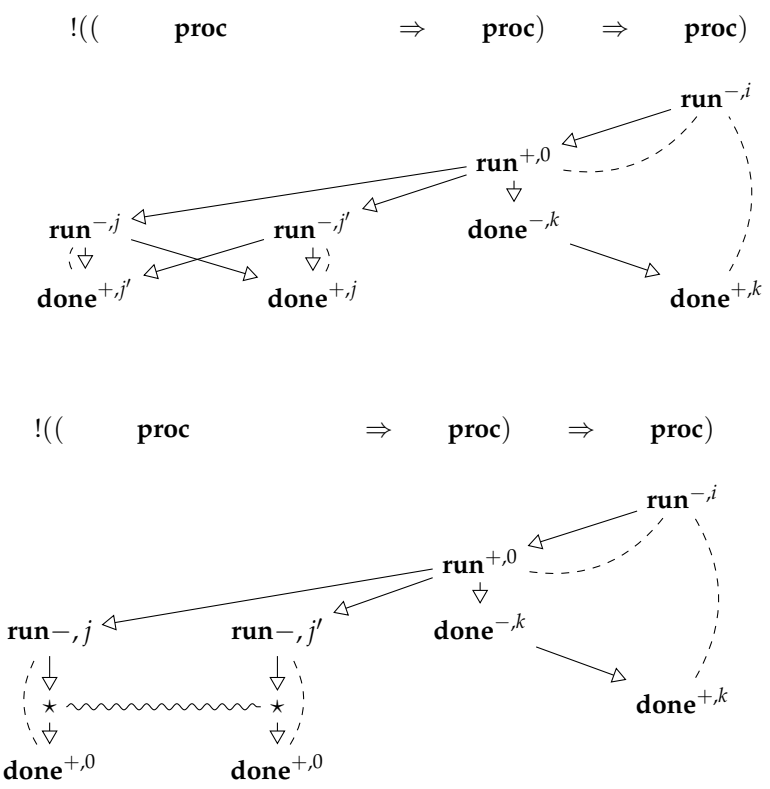


FIGURE 2. Interferences between independent branches

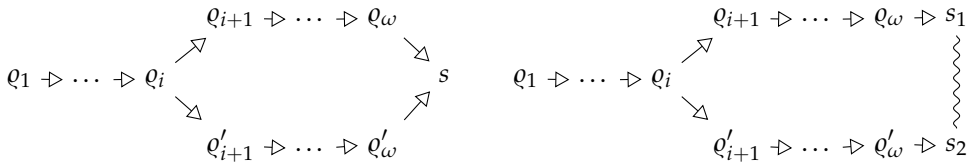
this kind of interference, one using causality, the other conflict. Intuitively, similar behaviours are exhibited by the following IPA terms:

$$\begin{aligned} M_{\to} = \lambda f^{\mathbf{proc}\Rightarrow\mathbf{proc}}.\ & \text{new } r \text{ in} \\ & f\,(\mathtt{incr}\, r; \text{wait } (r \geq 2)) \\ M_{\sharp} = \lambda f^{\mathbf{proc}\Rightarrow\mathbf{proc}}.\ & \text{new } r \text{ in} \\ & f\,(\mathtt{incr}\, r; \text{wait } (r = 1)) \end{aligned}$$

In $M_{\to}$, f is called with an argument that can only converge when f calls it at least twice concurrently. As a result, $M_{\to}$ diverges on $\lambda x.\, x$ and $\lambda x.\, x; x$ but converges on $\lambda x. x \parallel x$. Similarly in $M_{\sharp}$, only one call to the argument of f will converge. Note that $\lambda x.\, x \parallel x$ and $\lambda x.\, x$ cannot be distinguished by pure terms of IPA (for either may or must), but are distinguished by $M_{\to}$ for may-equivalence and by $M_{\sharp}$ for must-equivalence.

It is to be noted that the descriptions of Figure 2 are ambiguous: there might be several strategies in which our diagrams can embed. Since our discussion is at an informal level, we do not dwell on this issue, but we will show that for innocent strategies this representation is non-ambiguous (Section 1 of Chapter 6).

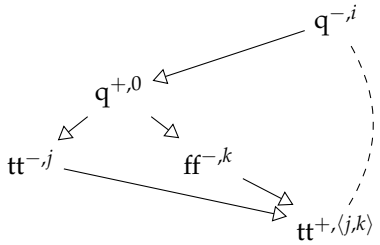
2.1.2. *Tracking forks via gccs.* To formulate our innocence condition, we need a bit of terminology about gccs. Let $\sigma : S \rightharpoonup A \in \mathrm{nCG}^{\cong}_{\circledcirc}$. Two gccs $\varrho, \varrho' \in \mathrm{gcc}(S)$ are **consistent** when $\varrho \cup \varrho' \in \mathrm{Con}_S$. Two consistent gccs are **forking** if there exists $n \in \mathbb{N}$ such that $\varrho_i = \varrho'_i$ for $i < n$, and $\varrho_{\geq n} \cap \varrho'_{\geq n} = \emptyset$. Intuitively, they coincide until a certain point (n) from which they are disjoint. In particular ($n = 0$), disjoint gccs are forking. If defined, ϱ_n is the **forking point**. They are **O-forking** if $n = 0$ or $\varrho_{n-1} = \varrho'_{n-1}$ is positive, and **P-forking** otherwise. Two forking gccs ϱ, ϱ' are **joined by** s if $\varrho_\omega \rightarrowtriangle s$ and $\varrho'_\omega \rightarrowtriangle s$, and **are racing** at s_1, s_2 if $\varrho_\omega \rightarrowtriangle s_1$ and $\varrho_\omega \rightarrowtriangle s_2$ with $s_1 \smile s_2$. In picture, joined gccs (left) and racing gccs (right):



We say that two forking gccs are **interfering** if they are joined or racing.

2.1.3. *Preinnocence.* Using this terminology, we can define preinnocence. A strategy in $\mathrm{nCG}^{\cong}_{\circledcirc}$ is **preinnocent** if its *O-forking* gccs are never interfering. For convenience, this condition will be split into two parts: a strategy is **$\rightarrowtriangle$-preinnocent** when two O-forking gccs are never joined and **$\sharp$-preinnocent** when two O-forking gccs are never racing. Unfortunately preinnocence is **not** stable under composition. Consider the preinnocent strategy `bad` of Figure 3. Precomposing it by $\alpha_{!\mathbb{B}}$ adds the causal dependence $\mathrm{tt}^{+,m_1} \rightarrowtriangle \mathrm{tt}^{-,j_1}$, resulting in the following strategy:

$$\texttt{bad} \circledcirc \alpha_{!\mathbb{B}} : \qquad !\mathbb{B} \qquad \Rightarrow \qquad \mathbb{B}$$



which is not $\rightarrowtriangle$-innocent since O-forking gccs are joined at $\mathrm{tt}^{+,\langle j,k\rangle}$ (a similar counter-example can be produced for $\sharp$-innocence).

A possible syntactic reading of this is as follows. Preinnocence can be approximated syntactically by restricting to terms of IPA that do not contain a read and a write instruction on the same reference in "two Opponent branches", which can be branches of the same conditionals or arguments of the same function call. The behaviour of the strategy `bad` can be defined in IPA by:

$$\begin{aligned}&\lambda f b.\, \mathrm{new}\ r, s\ \mathrm{in}\\ &\quad (\mathtt{if}\ (f\ (\mathrm{wait}\,(r = 1)))\ (s := 1; \bot)\ \bot)\\ &\quad ||\,(\mathtt{if}\ b\,(r := 1; \bot)\,(\mathrm{wait}\,(s = 1)))\end{aligned}$$

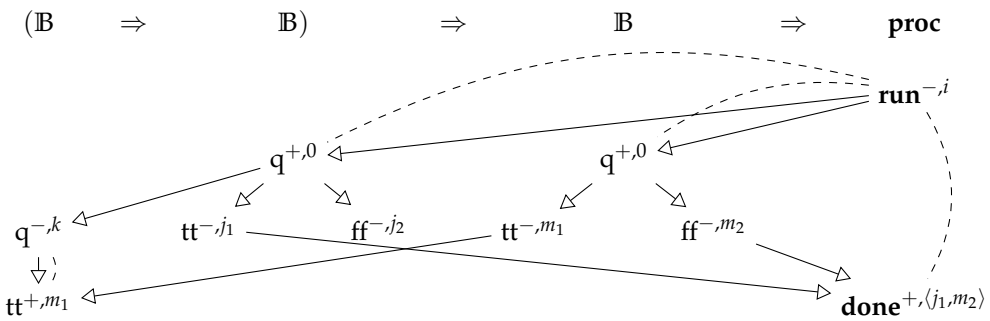


FIGURE 3. The strategy `bad`

If we put $f = \lambda x.\, x\ \mathrm{tt}$ we get a term equivalent to:

$$\begin{aligned}&\lambda b.\,\mathrm{new}\ r, s\ \mathrm{in}\\ &\quad(\mathrm{wait}\ (r = 1); s := 1; \bot)\\ &\quad||\ (\mathtt{if}\ b\ (r := 1; \bot)\ (\mathrm{wait}\ (s = 1)))\end{aligned}$$

which can be shown equivalent to:

$$\begin{aligned}&\lambda b.\,\mathrm{new}\ r, s\ \mathrm{in}\\ &\quad(\mathtt{if}\ b\ (r := 1; \bot)\ (\mathrm{wait}\ (r = 1)))\end{aligned}$$

in which there are operations of reading and writing on the same reference in branches of the same conditional. The analysis here is that the two parallel branches of `bad` (via the references r and s) interfere in a complex way. It is not possible to police the interferences between conditionals branches without *also* policing the interferences between different threads, leading to *locality*.

2.2. Locality. To solve this issue, we need also to rule out the strategy `bad`. For that, one can notice that there are causal links between two concurrent computations (two different threads).

This behaviour can be ruled out by the condition of **visibility** introduced in **[CCW15]**. It amounts to asking that each thread (seen as a sequential substrategy) be a valid strategy, in particular that its image in the game be downclosed:

DEFINITION 5.17. If A is an arena, a partial map of event structures $\sigma : S \rightharpoonup A$ is **visible** when for every gcc $\varrho \in \mathrm{gcc}(S)$, the set $\sigma\varrho = \{\sigma\varrho_0, \ldots, \sigma\varrho_\omega\}$ is a configuration of A.

As noticed in **[CCW15]**, visibility is stable under composition. Moreover, in the presence of visibility, $\rightarrowtriangle$-preinnocence is stable under composition. We prove both results in the next section, along with stability under composition of $\sharp$-preinnocence in the presence of visibility.

Visibility has an important characterisation in terms of justifiers:

LEMMA 5.18. *Let A an arena, and $\sigma : S \rightharpoonup A$ a partial map of event structures. It is visible if and only if for all non-minimal $s \in S_\downarrow$, and a gcc ϱ ending at s, ϱ contains the justifier of s.*

Note that this is the justifier as defined in Chapter 3 – not the extended justifier defined in Section 0, as σ is not assumed to be single-threaded.

PROOF. *only if.* If σ is visible, then $\sigma\varrho$ is a configuration of A hence it is down-closed in A. As a result ϱ must contain the justifier of s.

if. Let ϱ be a gcc and $s \in \varrho$ with $a \rightarrowtriangle \sigma s$. By assumption the justifier s' of s is in ϱ hence $a = \sigma s' \in \sigma_{\downarrow}\varrho$. □

This characterisation in terms of gccs sheds light on an interpretation of visibility as independence of concurrent computations. As a result, a player starting two concurrent threads is not allowed to add causal links between them without respecting the structure of types (the arenas), leading to another characterization:

LEMMA 5.19. *Let $\sigma : S \rightharpoonup A$ be a map of event structures to an arena. It is visible if and only if, for all comparable $s, s' \in S$, s and* $\mathrm{just}(s')$ *are comparable (whenever defined).*

PROOF. Assume σ is visible and let $s, s' \in S$ be comparable. If $s' \leq s$ then the conclusion follows from $\mathrm{just}(s') \leq s'$. If $s \leq s'$, assume that $\mathrm{just}(s')$ and s are concurrent. This means that we can find two gccs of s', one going through s and the other one through $\mathrm{just}(s)$. Hence the first one does not go through $\mathrm{just}(s')$ contradicting visibility of σ.

Conversely, assume σ satisfies the condition of the lemma and let ϱ be a gcc of S ending at s. All events of ϱ are comparable to s so they must be comparable to $\mathrm{just}(s)$. By transitivity, we must have two consecutive events ϱ_i and ϱ_{i+1} such that $\varrho_i \leq \mathrm{just}(s)$ and $\mathrm{just}(s) \leq \varrho_{i+1}$ (since $\varrho_0 \leq \mathrm{just}(s)$ and $\mathrm{just}(s) \leq \varrho_\omega = s$). Since $\varrho_i \rightarrowtriangle \varrho_{i+1}$, it follows that $\mathrm{just}(s) \in \varrho$ as desired. □

Through this characterisation of visibility, one can understand visibility as a constraint on causality with respect to justifiers (ie. with respect to the type). Following this intuition, we will also refer to visibility as $\rightarrowtriangle$**-locality**. Replacing causality by conflict in this characterization leads to $\sharp$-visibility:

DEFINITION 5.20. A partial map of event structure $\sigma : S \rightharpoonup A$ to an arena A is $\sharp$**-local** when for all $s \sharp s'$ in S then, if defined, s and $\mathrm{just}(s')$ are *not* concurrent, that is comparable or in conflict.

Say that an uncovered strategy is **local** when it is both $\rightarrowtriangle$-local and $\sharp$-local. Canonical non-local patterns are depicted in Figure 4. Note that $\sharp$-locality is a new condition not appearing in earlier publications. This condition is actually used once in the technical development of this thesis, to prove that, despite our weaker definition of well-bracketing, we recover all the conditions from **[CCW15]** when restricting to innocent and well-bracketed strategies, up to observational equivalence. Those conditions are key for finite definability. (See Proposition 6.15).

Locality can be restated in a compact way:

LEMMA 5.21. *A partial map of event structures $\sigma : S \rightharpoonup A$ to an arena A is local if and only if for $s, s' \in S$, if s is concurrent with* $\mathrm{just}(s')$ *then it is concurrent with s', whenever* $\mathrm{just}(s')$ *is defined.*

PROOF. A simple consequence of the definitions, since $\rightarrowtriangle$-locality is equivalent to: if s and s' are comparable, then s and $\mathrm{just}(s')$ are not concurrent (since they cannot be in conflict) and $\sharp$-locality to: if s and s' are in conflict then s and $\mathrm{just}(s')$ are not concurrent. □

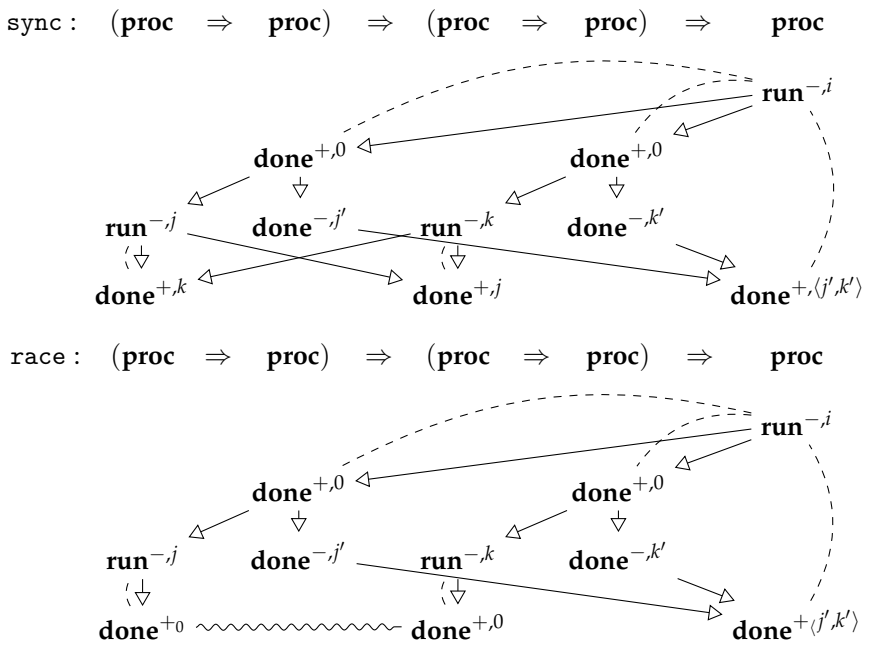


FIGURE 4. Interferences between concurrent subcomputations

For σ single-threaded and negative, the result works for extended justifiers:

LEMMA 5.22. *For an arena A and $\sigma : S \rightharpoonup A$ with S is negative and single-threaded,*

(1) σ is visible if and only if $s' \leq s$ implies that $\mathrm{just}_e(s)$ and s' are comparable
(2) σ is $\sharp$-local if and only if $s \sharp s'$ implies that $\mathrm{just}_e(s)$ and s' are not concurrent.
(3) σ is local if and only if, whenever $\mathrm{just}_e(s)$ and s' are concurrent, so are s and s'.

In these conditions, the quantification is over non-minimal s so that $\mathrm{just}_e(s)$ is defined.

PROOF. (3) follows directly from (2) and (1) by a similar reasoning as in Lemma 5.21. Conditions (1) and (2) follow from single-threadedness, we detail only (1).

Assume that $s \leq s'$. If $\mathrm{just}(s')$ is defined, the result follows from Lemma 5.19. If not, but $\mathrm{just}_e(s')$ is defined, this means that $\sigma s'$ is minimal and $\mathrm{just}_e(s')$ is the unique minimal event $\min(s')$ in $[s']$. Since $s \in [s']$, this implies $\min(s') \leq s$. □

In the rest of the document, we use *visibility* to refer to $\rightarrowtriangle$-locality as it has a central role (that $\sharp$-locality does not have) to show stability under composition of various conditions. The next section studies innocent strategies and proves that there is a CCC of innocent strategies.

3. Innocent strategies

The main technical challenge of this section is to establish that visibility, locality and innocence are stable under composition.

3.1. Interaction of visible strategies. The main steps towards this goal is to understand the interaction of visible strategies. Visibility seems to be quite a simple condition but it has strong repercussions on the concurrency patterns allowed

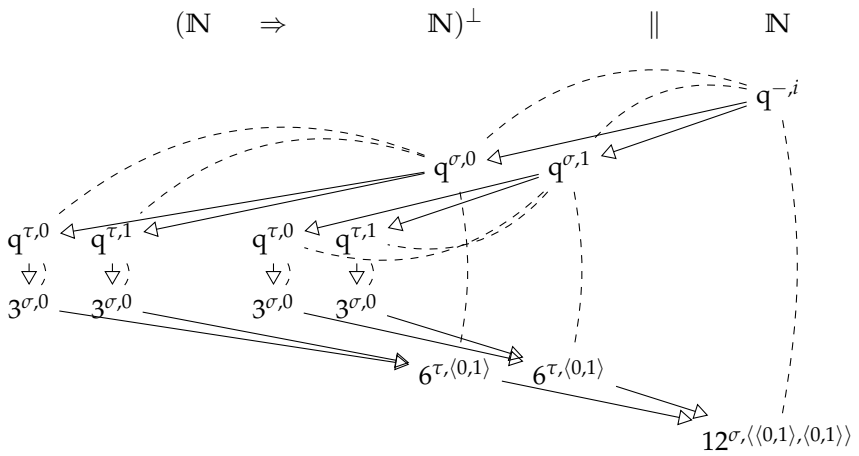


FIGURE 5. Interaction of $\sigma = [\![\lambda f.\, f\,3 + f\,3]\!]$ against $\tau = [\![\lambda x.\, x + x]\!]$

as evidenced by Lemma 5.29 to come. Throughout this section, we mainly consider two visible and courteous partial maps of event structures $\sigma : S \rightharpoonup A$ and $\tau : T \rightharpoonup A^{\perp}$ where A is an arena, and their interaction $\sigma \wedge \tau : S \wedge T \rightharpoonup A$. We make no further assumption than courtesy at the moment, as it is already enough to unravel interesting structure.

3.1.1. *Views in the interaction.* As a first step to show that visible strategies are stable under composition, we would like to show that $\sigma \wedge \tau$, as a partial map, is visible. However, given a gcc ϱ of $S \wedge T$, $\Pi_1\varrho$ and $\Pi_2\varrho$ have no reason to be gccs in general, making it hard to leverage our assumption on σ and τ.

EXAMPLE 5.23. Consider Figure 5 depicting the interaction of $\sigma = [\![\lambda f.\, f\,3 + f\,3]\!]$ against $\tau = [\![\lambda x.\, x + x]\!]$ with a concurrent semantics for $+$. Moves of the interaction are not annotated with the usual polarities $+, -$ but with the classification of moves in an interaction: σ-actions, τ-actions, and external Opponent (-). We have also drawn the extended justifier, well-defined as $\tau \circledast \sigma$ is negative and single-threaded. Consider the "leftmost" gcc of the interaction:

$$\varrho = \mathrm{q}^{-,i} \twoheadrightarrow \mathrm{q}^{\sigma,0} \twoheadrightarrow \mathrm{q}^{\tau,0} \twoheadrightarrow 3^{\sigma,0} \twoheadrightarrow 6^{\tau,-} \twoheadrightarrow 12^{\sigma,-}.$$

Because immediate causal links are inherited alternatively from σ and from τ, ϱ does not project to a gcc in S or $T \parallel !\mathbb{N}$. However, if we look at the events that are smaller than $12^{\sigma,-}$ in S, we get the following sub-sequence of ϱ:

$$\mathrm{q}^{-,i} \cdot \mathrm{q}^{\sigma,0} \cdot 6^{\tau,-} \cdot 12^{\sigma,-},$$

which is a *not* a gcc of the interaction, but once projected to σ becomes a gcc.

Formalizing this idea leads to the notion of views:

DEFINITION 5.24 (Views). If $\varrho \in \mathrm{gcc}(S \wedge T)$ such that $\Pi_1\varrho_\omega$ is defined (resp. $\Pi_2\varrho_\omega$ is defined), its **σ-view** (resp. **τ-view**) is the sub-sequence $\lceil\varrho\rceil^\sigma$ (resp. $\lceil\varrho\rceil^\tau$) containing the ϱ_k such that $\Pi_1\varrho_k \leq \Pi_1\varrho_\omega$ (resp. $\Pi_2\varrho_k \leq \Pi_2\varrho_\omega$).

We can show that the views actually do project to gccs:

LEMMA 5.25. *Let $\varrho \in \mathrm{gcc}(S \wedge T)$ such that $\Pi_1\varrho_\omega$ and $\Pi_2\varrho_\omega$ are both defined. If $\Pi_1\varrho_\omega$ (resp. $\Pi_2\varrho_\omega$) is defined, then $\Pi_1\lceil\varrho\rceil^\sigma$ (resp. $\Pi_2\lceil\varrho\rceil^\tau$) is a gcc of S (resp. of T)*

PROOF. We prove the result for both projections, by induction on ϱ. If ϱ is a singleton, then this is trivial. Assume $\varrho = \varrho' \cdot e$. There are several cases:

(1) $\Pi_1 e$ is nonnegative: write $e' := \varrho'_\omega$. By assumption we have $e' \twoheadrightarrow e$. By courtesy – since $\Pi_2 e$ is either undefined or negative, $\Pi_1 e' \twoheadrightarrow \Pi_1 e$ (in particular $\Pi_1 e'$ is defined). This, with ϱ being a linear order, induces the following equivalence for all $\varrho_i \in \varrho$

$$\Pi_1\varrho_i \leq \Pi_1 e \qquad \text{iff} \qquad \Pi_1\varrho_i = e \vee \Pi_1\varrho_i \leq e'.$$

This gives $\lceil\varrho\rceil^\sigma = \lceil\varrho'\rceil^\sigma \cdot e$, and we conclude by induction on ϱ'.

(2) $\Pi_2 e$ is nonnegative: similar reasoning as for (1)

(3) $\Pi_1 e$ is negative: then $\Pi_2 e$ is positive and by (2), $\Pi_2\lceil\varrho\rceil^\tau$ is a gcc. By visibility of τ, $\mathrm{just}_{\mathrm{e}}(\Pi_2 e) \in \Pi_2\lceil\varrho\rceil^\tau$. Since Π_2 preserves justifiers, we have $\Pi_2 \mathrm{just}_{\mathrm{e}}(e) = \mathrm{just}_{\mathrm{e}}(\Pi_2 e) \in \Pi_1\lceil\varrho\rceil^\sigma$. As a result, $\mathrm{just}_{\mathrm{e}}(e) \in \lceil\varrho\rceil^\sigma$ by local injectivity and $\mathrm{just}_{\mathrm{e}}(e) \in \varrho$. Since $\Pi_1(\mathrm{just}_{\mathrm{e}}\, e) = \mathrm{just}_{\mathrm{e}}(\Pi_1 e) \leq \Pi_1 e$, it follows that $\mathrm{just}_{\mathrm{e}}\, e \in \Pi_1\lceil\varrho\rceil^\sigma$. We have again for $\varrho_i \in \varrho$:

$$\Pi_1\varrho_i \leq \Pi_1 e \qquad \text{iff} \qquad \Pi_1\varrho_i = e \vee \Pi_1\varrho_i \leq e',$$

since $\Pi_1(\mathrm{just}_{\mathrm{e}}\, e) \twoheadrightarrow \Pi_1 e$ ($\Pi_1 e$ being negative). This shows that $\lceil\varrho\rceil^\sigma = \lceil\varrho_{\leq k}\rceil^\sigma \cdot e$ where k is the index of $\mathrm{just}_{\mathrm{e}}(e)$ in ϱ, which implies it is a gcc. □

The proof have made use of the following equations:

$$\lceil\varrho \cdot e\rceil^\sigma = \lceil\varrho\rceil^\sigma \cdot e \qquad (\Pi_1 e \text{ nonnegative})$$

$$\lceil\varrho \cdot \mathrm{just}(e) \cdot \ldots \cdot e\rceil^\sigma = \lceil\varrho \cdot \mathrm{just}(e)\rceil^\sigma \cdot e \qquad (\Pi_1 e \text{ negative, non-minimal})$$

The reader familiar with HO game semantics may recognize the definition of views in traditional HO game semantics. Indeed if σ and τ are deterministic and sequential, this coincides with the standard definition.

The previous lemma implies that:

LEMMA 5.26. *Let $\varrho \in \mathrm{gcc}(T \circledast S)$ with at least two moves, such that $\Pi_1\varrho_\omega$ is defined. Write ϱ_r for the predecessor of ϱ_ω in $\lceil\varrho\rceil^\sigma$. Then:*

- *If $\Pi_1\varrho_\omega$ is nonnegative, then ϱ_r is the predecessor of ϱ_ω in ϱ (ie. $\varrho_r \twoheadrightarrow \varrho_\omega$)*
- *If $\Pi_1\varrho_\omega$ is negative, then ϱ_r is the extended justifier of ϱ_ω.*

PROOF. Straightforward by visibility and Lemma 5.25. □

From this construction we deduce that visibility is stable under composition.

LEMMA 5.27. *Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be linear visible strategies. Then $\tau \circledcirc \sigma$ is a visible strategy.*

PROOF. We use the characterisation of visibility in terms of justifiers (Lemma 5.18). Let $\varrho \in \mathrm{gcc}(T \circledcirc S)$ ending in visible e such that $(\tau \circledcirc \sigma)e$ is not minimal. By filling in the gaps, this gcc induces (non-uniquely) a gcc $\varrho' \in \mathrm{gcc}(T \circledast S)$ that contains ϱ, ending at e. Since e is visible, $\mathrm{just}(\Pi_1 e) \in \Pi_1\lceil\varrho'\rceil^{\sigma\parallel C}$. This means that $\mathrm{just}(e) \in \varrho'$ and since $\mathrm{just}(e)$ is visible, $\mathrm{just}(e) \in \varrho$. □

3.1.2. *Forking lemma.* Innocence and locality can be seen as a way to control interferences between forking gccs. To show that they are stable under composition, we must, from a fork of an interaction of innocent strategies go back to a fork of one of the strategies. Views allow us to extract gccs of the strategies from gccs of their interaction. In this section, we show that this process preserves enough good properties about the forking structure of the strategies. Throughout this section, we consider $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ visible strategies of $\mathrm{nCG}^{\cong}_{\circledcirc}$. Write $\sigma' := \sigma \parallel C$ (with $S' := S \parallel C$), and $\tau' := A \parallel \tau$ (with $T' := A \parallel T$).

First, the operation $\varrho \mapsto \Pi_1 \lceil \varrho \rceil^\sigma$ is not monotonic with respect to the prefix ordering on gccs. However if χ is a prefix of ϱ but $\Pi_1 \lceil \chi \rceil^\sigma$ is not a prefix of $\Pi_1 \lceil \varrho \rceil^\sigma$, then they are O-forking. Indeed, if you consider the gcc ϱ of Example 5.23, it is obvious to see that the view (for σ) of the prefix

$$\chi = \mathrm{q}^{-,0} \cdot \mathrm{q}^{\sigma,0} \cdot \mathrm{q}^{\tau,0} \cdot 3^{\sigma,0}$$

is itself. However, in this case, their forking point is $\mathrm{q}^{\sigma,0}$ which is *positive* for σ. This is always the case:

LEMMA 5.28. *Let ϱ, χ be gccs of $T \circledast S$ with visible maximal events and $\chi \subseteq \varrho$. If $\chi_\omega \notin \lceil \varrho \rceil^{\sigma'}$ then $\Pi_1 \lceil \chi \rceil^{\sigma'}$ and $\Pi_1 \lceil \varrho \rceil^{\sigma'}$ are O-forking gccs.*

Note in passing that $\chi_\omega \in \lceil \varrho \rceil^{\sigma'}$ if and only if $\lceil \chi \rceil^{\sigma'} \subseteq \lceil \varrho \rceil^{\sigma'}$.

PROOF. We proceed by induction on the sum of sizes of $\lceil \varrho \rceil^{\sigma'}$ and $\lceil \chi \rceil^{\sigma'}$. If $\lceil \varrho \rceil^{\sigma'}$ is a singleton, then $\lceil \varrho \rceil^{\sigma'}$ and $\lceil \chi \rceil^{\sigma'}$ must be disjoint by assumption on χ_ω and O-forking (disjoint gccs are O-forking by definition). Moreover, if $\varrho = \chi$ then, the result is obvious as well so we assume $\chi_\omega < \varrho_\omega$.

Consider the predecessor ϱ_r of ϱ_ω in $\lceil \varrho \rceil^{\sigma'}$. If $\varrho_r \geq \chi_\omega$, then $\chi \subseteq \varrho_{\geq r}$ and we can apply the induction hypothesis.

The most interesting case is $\varrho_r < \chi_\omega$ (which is only possible if $\lceil \chi \rceil^{\sigma'}$ is not a subset of $\lceil \varrho \rceil^{\sigma'}$). We have the following situation:

$$\varrho_0 \mathrel{\triangleright\!\!\!\cdot} \cdots \mathrel{\triangleright\!\!\!\cdot} \varrho_r \overset{\frown}{\mathrel{\triangleright\!\!\!\cdot}} \cdots \mathrel{\triangleright\!\!\!\cdot} \chi_\omega \mathrel{\triangleright\!\!\!\cdot} \cdots \mathrel{\triangleright\!\!\!\cdot} \varrho_\omega$$

Since $\varrho_r \not\twoheadrightarrow \varrho_\omega$, but ϱ_r is the predecessor of ϱ_ω in $\lceil \varrho \rceil^{\sigma'}$, it must be that $\Pi_1 \varrho_\omega$ is negative and $\Pi_1 \varrho_r$ positive (by Lemma 5.26). In this case, if $\varrho_r \in \lceil \chi \rceil^{\sigma'}$, then ϱ and χ are forking, and the forking point is ϱ_r which is positive as desired. □

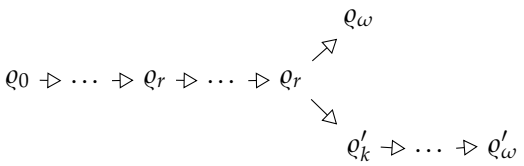
LEMMA 5.29 (Forking Lemma). *Assume we have two gccs ϱ, ϱ' of $T \circledast S$ forking at e. If $\Pi_1 \lceil \varrho \rceil^\sigma$ and $\Pi_1 \lceil \varrho' \rceil^\sigma$ are not forking at $\Pi_1 e$, then they are O-forking.*

PROOF. We proceed by induction on the length of ϱ. Let ϱ_r be the predecessor of ϱ_ω in $\lceil \varrho \rceil^{\sigma'}$ We write χ for $\varrho_{\leq r}$. If $\varrho_r > e$, then χ and ϱ are forking at $\Pi_1 e$ as well, and we can conclude by induction hypothesis since

$$\lceil \varrho \rceil^{\sigma'} = \lceil \chi \rceil^{\sigma'} \cup \{\varrho_\omega\}.$$

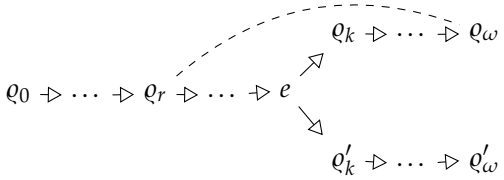
The hard case is when $\varrho_r \leq e$, and we have $\chi \subseteq \varrho'$. There are two sub-cases:

- If ϱ_ω is nonnegative so that $\varrho_r \twoheadrightarrow \varrho_\omega$. In that case, $\varrho_r = e$, and we have:



In this case, by Lemma 5.28, either $\Pi_1\lceil\chi\rceil^{\sigma'} \subseteq \Pi_1\lceil\varrho'\rceil^{\sigma'}$, or $\Pi_1\lceil\chi\rceil^{\sigma'}$ and $\Pi_1\lceil\varrho'\rceil^{\sigma'}$ are O-forking. The first condition implies that $\Pi_1\lceil\varrho\rceil^{\sigma'}$ and $\Pi_1\lceil\varrho'\rceil^{\sigma'}$ are forking at $\Pi_1\varrho_r = \Pi_1 e$, and the second one that they are O-forking.

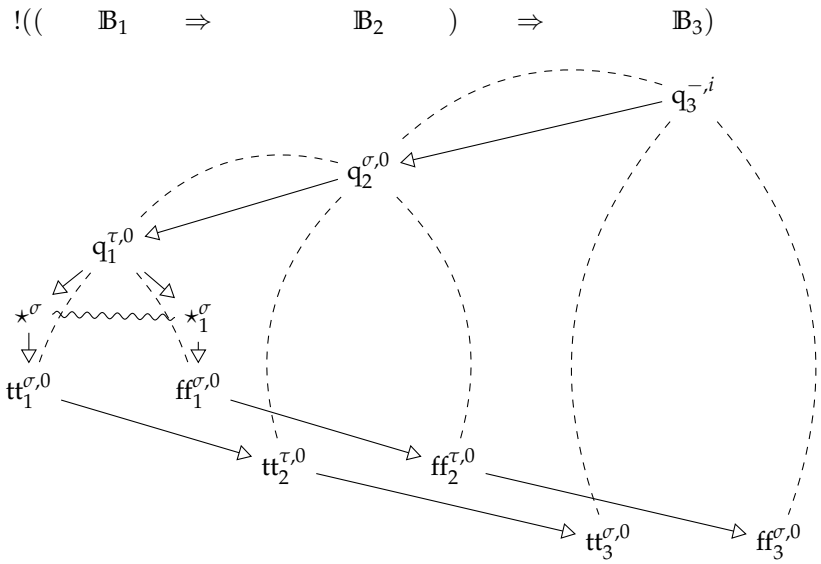
- If ϱ_ω is negative, then we have the following picture:



Similarly, since $\chi \subseteq \varrho'$, either $\lceil\chi\rceil^{\sigma'} \subseteq \lceil\varrho\rceil^{\sigma'}$ or $\Pi_1\lceil\varrho\rceil^{\sigma'}$ and $\Pi_1\lceil\varrho'\rceil^{\sigma'}$ are O-forking (by Lemma 5.28). Both conditions imply that $\Pi_1\lceil\varrho\rceil^{\sigma'}$ and $\Pi_1\lceil\varrho'\rceil^{\sigma'}$ are O-forking, as desired. □

As a consequence, if the gccs in the interaction are O-forking (from a point of view of σ' or τ'), the corresponding gccs in σ' or τ' will also be O-forking. This argument is key to prove stability under composition of innocence.

3.1.3. *Stability under composition of locality.* We now move on to stability under composition of locality. To prove this, we hit a similar problem as for visibility: a conflict in the interaction might not project to a conflict in the strategies. For instance, consider the interaction of $\sigma = [\![\lambda f.f\ \texttt{choice}]\!]$ against $\tau = \mathfrak{a}_{\mathbb{B}}$:



The events tt_3 and ff_3 are in conflict in the interaction (inherited from tt_1 and ff_1), even though in copycat those moves are concurrent. We remark that in the views (both for σ and τ) of tt_3, tt_1 is absent: the conflict is invisible, hence there is no way of appealing to locality of σ or τ here. However, we still have that the justifier of tt_3 (the initial question) is smaller than ff_3 so that locality is not violated.

This intuition is captured by the following lemma:

LEMMA 5.30. *Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be local strategies of $\mathrm{nCG}^{\cong}_{\circledcirc}$. Consider the following situation in $T \circledast S$:*

$$\varrho_0 \dashrightarrow \cdots \dashrightarrow \varrho_k \dashrightarrow \begin{array}{l} e_1 \dashrightarrow \cdots \dashrightarrow e \\ \wr \\ e_2 \end{array}$$

If $e_1 \notin \lceil \varrho \rceil^{\sigma'}$ (resp. $e_1 \notin \lceil \varrho \rceil^{\tau'}$) then no moves in $\lceil \varrho \rceil^{\sigma'}$ (resp. $\lceil \varrho \rceil^{\tau'}$) are concurrent to e_2.

PROOF. We prove the result for both strategies by mutual induction on such gccs. Write $\varrho = \varrho' \cdot e$ and write e' for ϱ'_ω. In particular, we have $e > e_1$. In the proof we write $\lceil d \rceil^{\sigma'}$ when $d \in \varrho$ for $\lceil \varrho \rceil^{\sigma'} \cap [d]$, and similarly for $\lceil d \rceil^{\tau'}$.

If $\Pi_1 e$ is nonnegative: in this case, $\lceil e \rceil^{\sigma'} = \lceil e' \rceil^{\sigma'} \cdot e$ and we can conclude by induction since e is not concurrent to e_2 and $e' \geq e_1$.

If $\Pi_2 e$ is nonnegative: same reasoning.

If $\Pi_1 e$ is negative: If $\mathrm{just}(e) \geq e_1$, we can conclude by induction since

$$\lceil e \rceil^{\sigma'} = \lceil \mathrm{just}(e) \rceil^{\sigma'} \cdot e.$$

Otherwise, we cannot simply use the induction hypothesis. We show that $\mathrm{just}(e)$ is not concurrent to e_2, which implies $\mathrm{just}(e) \leq e_2$ by minimality of the conflict $e_1 \rightsquigarrow e_2$. This in turn implies that $\lceil \mathrm{just}(e) \rceil^{\sigma'} \subseteq [e_2]$ and implies the result.

If $e_1 \notin \lceil e \rceil^{\tau'}$, then since $\mathrm{just}(e) \in \lceil e \rceil^{\tau'}$, it is not concurrent to e_2 by the previous point. If $e_1 \in \lceil e \rceil^{\tau'}$, then we apply locality of τ' as in this case $\Pi_2 e \geq \Pi_2 e_1$ is conflict with $\Pi_2 e_2$, which directly gives that $\mathrm{just}(\Pi_2 e)$ is not concurrent to $\Pi_2 e_2$.

If $\Pi_2 e$ is negative: same reasoning. □

LEMMA 5.31. *Let $\sigma : S \to A^\perp \parallel B$ and $\tau : T \to B^\perp \parallel C$ be local strategies of $\mathrm{nCG}^{\cong}_{\circledcirc}$. Then $\tau \circledcirc \sigma$ is local.*

PROOF. By Lemma 5.27, $\tau \circledcirc \sigma$ is visible. We establish $\sharp$-locality.

Let e and e' be conflicting event of $T \circledcirc S$. By definition of the hiding and minimal conflict, we can find $e_1, e_2 \in T \circledast S$ with $e_1 \leq e$, $e_2 \leq e'$ and $e_1 \rightsquigarrow e_2$. Consider a gcc of e going through e_1.

If its σ'-view contains e_1, then by locality of σ, $\mathrm{just}(e)$ is not concurrent to e_2 as desired. Otherwise, by Lemma 5.30, $\lceil \varrho \rceil^{\sigma'}$ does not contain any move concurrent to e_2, so in particular $\mathrm{just}(e)$ is not concurrent to e_2. □

3.1.4. *Deadlock-free lemma.* A crucial property about the interaction of visible strategies is that all visible strategies have deadlock-free interactions.

We first prove the lemma for *dual* visible uncovered strategies, on a game A with only *negative* minimal events. So consider visible $\sigma : S \rightharpoonup A$ (necessarily negative), and $\tau : T \rightharpoonup A^\perp$ (non-negative). We assume moreover that events in S (resp. T) that map to minimal events of A are minimal.

In such a situation, we have:

LEMMA 5.32. *In a situation as above, for any $x \in \mathscr{C}(S), y \in \mathscr{C}(T)$ such that $\sigma\, x = \tau\, y$, the bijection*

$$\varphi : x \parallel y_\star \simeq x_\star \parallel \sigma x \parallel y_\star = x_\star \parallel \tau y \parallel y_\star \simeq x_\star \parallel y,$$

induced by local injectivity, is secured.

PROOF. Observe first that because $\sigma s = \tau(\varphi(s))$, it follows that φ preserves justifiers: $\varphi(\mathrm{just}(s)) = \mathrm{just}(\varphi s)$. We recall that φ is secured when the relation $(s,t) \lhd_\varphi (s',t')$ defined on the graph of φ as $s <_S s'$ or $t <_T t'$ is acyclic. Suppose it is not, and consider a cycle $((s_1,t_1),\dots,(s_n,t_n))$ with

$$(s_1,t_1) \lhd_\varphi (s_2,t_2) \lhd_\varphi \dots \lhd_\varphi (s_n,t_n) \lhd_\varphi (s_1,t_1)$$

Let us first give a measure on such cycles. The **length** of a cycle as above is n. For $a \in A$, the **depth**, $\mathrm{depth}(a)$, of a is the length of the path to a minimal event of the arena – so the depth of a minimal event is 0. Then, the **depth** of the cycle above is the sum:

$$d = \sum_{1 \le i \le n} \mathrm{depth}(\sigma\, s_i)$$

Cycles are well-ordered by the lexicographic ordering on (n,d); let us now consider a cycle which is minimal for this well-order. Note: in this proof, all arithmetic computations on indices are done modulo n (the length of the cycle).

Since $\le_S$ and $\le_T$ are transitive we can assume that $s_{2k} \le s_{2k+1}$ and $t_{2k+1} \le t_{2k+2}$ for all k. This shows that all involved events must be visible. It follows by minimality that $\mathrm{pol}_S(s_{2k}) = -$ and $\mathrm{pol}_S(s_{2k+1}) = +$ so that the cycle is alternating. Indeed, assume

$$(s_{2k+1},t_{2k+1}) \lhd_\varphi (s^+_{2k+2},t^-_{2k+2}) \lhd_\varphi (s_{2k+3},t_{2k+3})$$

with $t_{2k+1} \le_T t_{2k+2}$ and $s_{2k+2} \le_S s_{2k+3}$. The causal dependency $t_{2k+1} \le_T t^-_{2k+2}$ decomposes into $t_{2k+1} \le_T t \rightarrowtriangle_T t^-_{2k+2}$, with by courtesy $\tau\, t \rightarrowtriangle_A \tau\, t_{2k+2}$. Note that as A is alternating, this entails that $\mathrm{pol}_T(t) = +$. There must be some $(s,t) \in \varphi$, with $\mathrm{pol}_S(s) = -$. But since $\sigma\, s \le_A \sigma\, s_{2k+2}$, we must have $s \le_S s_{2k+2}$ as well, therefore we can replace the cycle fragment above with

$$(s_{2k+1},t_{2k+1}) \lhd_\varphi (s^-,t^+) \lhd_\varphi (s_{2k+3},t_{2k+3})$$

which has the same length but smaller depth, absurd. By the dual reasoning, events with odd index must have polarity as in (s^+_{2k+1},t^-_{2k+1}) as well.

Now, we remark that the cycle cannot contain events that are minimal in the game. Indeed, by hypothesis a synchronized event (s,t) such that $\sigma\, s = \tau\, t \in A$ is minimal in A is such that $s \in S$ and $t \in T$ are minimal as well, so (s,t) is a root for $\lhd_\varphi$ and cannot be in a cycle. Therefore, all events in the cycle have a predecessor in the game, *i.e.* a justifier.

Since $s_{2k} <_S s_{2k+1}$, by Lemma 5.19, $\mathrm{just}(s_{2k+1})$ is comparable with s_{2k} in S. They have to be distinct, as otherwise we would have $\sigma s_{2k} \rightarrowtriangle_A \sigma s_{2k+1}$ which in turn implies $t_{2k} <_T t_{2k+1}$. This gives $t_{2k-1} <_T t_{2k+2}$ hence (s_k,t_k) and (s_{k+1},t_{k+1}) can be removed without breaking the cycle, contradicting its minimality. By a similar reasoning, $\mathrm{just}(t_{2k+2})$ is comparable and distinct from t_{2k+1}.

Assume that we have $s_{2k} < \mathrm{just}(s_{2k+1})$ for some k. Since $\mathrm{just}(s_{2k+1}) < s_{2k+1}$ and $\mathrm{just}(t_{2k+1}) < t_{2k+1} < t_{2k+2}$. Therefore, we can replace the cycle fragment

$$(s_{2k},t_{2k}) \lhd_\varphi (s_{2k+1},t_{2k+1}) \lhd_\varphi (s_{2k+2},t_{2k+2})$$

with the cycle fragment

$$(s_{2k}, t_{2k}) \lhd_\varphi (\mathrm{just}(s_{2k+1}), \mathrm{just}(t_{2k+1})) \lhd_\varphi (s_{2k+2}, t_{2k+2})$$

which has the same length but smaller depth, absurd. So we must have $\mathrm{just}(s_{2k+1}) < s_{2k}$. Similarly, we must have $\mathrm{just}(t_{2k+2}) < t_{2k+1}$ for all k.

So we have that for all k, $\mathrm{just}(s_{2k+1}) < s_{2k}$ with $\mathrm{pol}_S(s_{2k}) = -$. By courtesy and the fact that A is alternating, this has to factor as

$$\mathrm{just}(s_{2k+1}) <_S \mathrm{just}(s_{2k})^+ \rightarrowtriangle_S s_{2k}^-$$

By the dual reasoning, we have that $\mathrm{just}(t_{2k+2}) <_T \mathrm{just}(t_{2k+1})$, as

$$\mathrm{just}(s_{2k+1}) \neq \mathrm{just}(s_{2k}) \quad \text{and} \quad \mathrm{just}(t_{2k+1}) \neq \mathrm{just}(t_{2k+2})$$

since they have different polarities.

So we have proved that we always have $\mathrm{just}(s_{2k+1}) <_S \mathrm{just}(s_{2k})$ and $\mathrm{just}(t_{2k+2}) <_t \mathrm{just}(t_{2k+1})$. That means that we can replace the full cycle

$$(s_1, t_1) \lhd_\varphi (s_2, t_2) \lhd_\varphi \ldots \lhd_\varphi (s_n, t_n) \lhd_\varphi (s_1, t_1)$$

with the cycle

$$\begin{array}{l}(\mathrm{just}(s_1), \mathrm{just}(t_1)) \lhd_\varphi (\mathrm{just}(s_n), \mathrm{just}(t_n)) \lhd_\varphi \\ (\mathrm{just}(s_{n-1}), \mathrm{just}(t_{n-1})) \lhd_\varphi \cdots \lhd_\varphi (\mathrm{just}(s_1), \mathrm{just}(t_1))\end{array}$$

which has the same length but smaller depth, absurd. □

The lemma above is the core of the proof. However, some more bureaucratic reasoning is necessary to reduce an open interaction to this setting.

Consider $\sigma : S \to A^\perp \parallel B$ and $\tau : T \to B^\perp \parallel C$ which are both visible strategies of $\mathrm{nCG}^{\cong}_{\circledcirc}$, and write $\sigma_A : S \rightharpoonup A^\perp, \sigma_B : S \rightharpoonup B, \tau_B : T \rightharpoonup B$ and $\tau_C : T \rightharpoonup C$ for the component-wise maps. We cannot use transparently the lemma above, because the interaction of σ and τ involves the closed interaction of $\sigma \parallel C^\perp$ and $A \parallel \tau$ on the arena $A \parallel B^\perp \parallel C$, which is not negative.

However, we prove that any $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$, synchronized in the sense that $\sigma_B x = \tau_B y$ admit refinement x' and y' playing on $((A \Rightarrow B) \Rightarrow C)^\perp$ and $(A \Rightarrow B) \Rightarrow C$ respectively.

First, similarly as in CHO, we can turn any strategy of $\mathrm{nCG}^{\cong}_{\circledcirc}$ on $A^\perp \parallel B$ into a strategy on $A \Rightarrow B$ using single-threadedness.

LEMMA 5.33. *Let $\sigma : S \rightharpoonup A^\perp \parallel B$. The following function $\Lambda(\sigma) : S \rightharpoonup (A \Rightarrow B)$ defines a visible strategy in* $\mathrm{nCG}^{\cong}_{\circledcirc}$*:*

$$s \in x \mapsto \begin{cases} (1, \sigma s) & (\sigma s \in B) \\ (0, (\sigma(\min(s)), \sigma s)) & (\sigma s \in A) \\ \textit{undefined} & (\textit{otherwise, that is } \sigma s \textit{ undefined}) \end{cases}$$

PROOF. Straightforward verification. □

LEMMA 5.34. *Let $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$ such that $\sigma_B x = \tau_B y$. There exist partial orders $\overline{x}$ and $\overline{y}$ such that*

1. *the support of $\overline{x}$ is $x \parallel \tau_B y$ and the identity function defines a map of event structures $\overline{x} \to x \parallel \tau_B y$*
2. *there is a map of event structure $\overline{\sigma_x} : \overline{x} \rightharpoonup ((A \Rightarrow B) \Rightarrow C)^\perp$ which is a uncovered strategy,*

(3) *the support of* $\overline{y}$ *is* $\sigma_B x \parallel y$ *and the identity function defines a map of event structures* $\overline{y} \to \sigma_B x \parallel y$
(4) *There is a partial map of event structure* $\overline{\tau_y} : \overline{y} \rightharpoonup ((A \Rightarrow B) \Rightarrow C)$ *that turn* $\overline{y}$ *into a* negative*, and visible uncovered strategy*

PROOF. We remark that by single-threadedness of τ, to each $s \in x$ corresponds a minimal $\min_T(s) \in x$. Indeed, $\sigma_B s \in \sigma_B x = \tau_B y$ hence there exists $t \in y$ with $\tau_B t = \sigma_B s$. Then we set $\min_T(s) := \min(t)$.

Define $\bar{x}$ as the partial order $(x \parallel \tau_B y, \leq_{x'})$ where

$$\leq_{\bar{x}} = \leq_{x \parallel \tau_B y} \cup \{((1, \tau(\min{}_T(s)),\ (0, s)) \mid s \in x\}.$$

This definition satisfies condition (1). For condition (2) define a map $\bar{x} \to ((A \Rightarrow B) \Rightarrow C)^\perp$ as follows:

$$(0, s) \mapsto (0, (\tau(\min{}_T(s)), \Lambda(\sigma)(s)))$$
$$(1, c) \mapsto (1, c)$$

It is easily checked to satisfy the required properties.

Definition for $\bar{y}$ is similar. □

We can now conclude:

THEOREM 5.35 (Deadlock free lemma). *Let* $\sigma : S \rightharpoonup A^\perp \parallel B \in \mathrm{nCG}^{\cong}_{\circledcirc}(A, B)$ *and* $\tau : T \rightharpoonup B^\perp \parallel C \in \mathrm{nCG}^{\cong}_{\circledcirc}(B, C)$ *be visible strategies. Let* $x \in \mathscr{C}(S)$ *and* $y \in \mathscr{C}(T)$ *such that* $\sigma_B x = \tau_B y$*. Then there exists a (necessarily unique) configuration* $z \in \mathscr{C}(T \circledast S)$ *such that* $\Pi_1 z = x \parallel \tau_C y$ *and* $\Pi_2 z = \sigma_A x \parallel y$*.*

PROOF. Consider such x and y. From Lemma 5.34, build $\overline{\sigma_x} : \overline{x} \rightharpoonup ((A \Rightarrow B) \Rightarrow C)^\perp$ and $\overline{\sigma_y} : \overline{y} \rightharpoonup ((A \Rightarrow B) \Rightarrow C)$. Note that $\overline{\sigma_x}\overline{x} = \overline{\tau_y}\overline{y}$, as a result there is a bijection $\bar{x} \simeq \bar{y}$. Since those extended strategies satisfy conditions of Lemma 5.32, this bijection is secured. Since causal dependence is more constrained in $\bar{x}$ and $\bar{y}$ than in $x \parallel y_\star \parallel \tau_C y$ and $\sigma_A x \parallel x_\star \parallel y$ respectively, the deduced bijection

$$x \parallel y_\star \parallel \tau_C y \simeq \sigma_A x \parallel x_\star \parallel y$$

is secured. As a result, it induces the desired configuration $z \in \mathscr{C}(T \circledast S)$. □

3.2. Innocent strategies. To define innocence, we follow the intuitions given in Section 2.1:

DEFINITION 5.36 (Innocence). Let $\sigma : S \rightharpoonup A$ be an uncovered *local* strategy. It is innocent when it satisfies the following two conditions:

- (*$\rightarrowtriangle$-preinnocence*) Two O-forking gccs are never joined.
- (*$\sharp$-preinnocence*) Two O-forking gccs are never racing

3.2.1. *Innocence and the interaction.* Let $\sigma : S \rightharpoonup A^\perp$ and $\tau : T \rightharpoonup A$ be innocent uncovered strategies. Since σ and τ are visible, the forking lemma induces properties that will be key to prove that innocence is stable under composition:

LEMMA 5.37. *Let* ϱ, ϱ' *be forking gccs of* $T \circledast S$ *joined by* p *such that* $\Pi_1 p$ *is nonnegative in* S*. Then* $\Pi_1 p$ *joins the gccs* $\Pi_1 \lceil \varrho \rceil^\sigma$ *and* $\Pi_1 \lceil \varrho' \rceil^\sigma$ *(which are forking by Lemma 5.29), and similarly if* $\Pi_2 p$ *is nonnegative in* T *instead.*

PROOF. Since $\Pi_1 e$ is positive, we know that by Lemma 2.65 that $\varrho_\omega \twoheadrightarrow e$ implies $\Pi_1 \varrho_\omega \twoheadrightarrow \Pi_1 e$ by courtesy, hence $\Pi_1 \lceil \varrho \rceil^\sigma$ and $\Pi_1 \lceil \varrho' \rceil^\sigma$ are joined by $\Pi_1 e$. □

LEMMA 5.38. *Let ϱ, ϱ' be forking gccs of $T \circledast S$ racing at e and e'. In particular, $\Pi_1 e$ and $\Pi_1 e'$ are internal for σ or for τ. Assume they are internal for σ, then $\Pi_1 \lceil \varrho \rceil^\sigma$ and $\Pi_1 \lceil \varrho' \rceil^\sigma$ are racing at $\Pi_1 e$ and $\Pi_1 e'$ (well defined once again by Lemma 5.29)*

PROOF. We have $\varrho_\omega \twoheadrightarrow \Pi_1 p$ and $\varrho'_\omega \twoheadrightarrow \Pi_1 p'$ because $\Pi_1 p$ and $\Pi_1 p'$ are internal moves of S. By Lemma 2.65, they must be in conflict and this conflict must be minimal because Π_1 reflects conflict. □

3.2.2. *The category* $\mathrm{CHO}_{\mathtt{inn}}$. The two previous lemmata allow us to conclude that innocence is stable under composition.

LEMMA 5.39. *Innocent strategies of* $\mathrm{nCG}^{\cong}_{\circledcirc}$ *are stable under composition.*

PROOF. Let $\sigma : S \rightharpoonup A^\perp \parallel B$ and $\tau : T \rightharpoonup B^\perp \parallel C$ be innocent linear strategies. We already know that $\tau \circledcirc \sigma$ is local.

- ($\rightarrowtriangle$-*innocence*) Consider two O-forking gccs ϱ, ϱ' of $T \circledcirc S$ that are joined at e. We can complete them into gccs $\bar{\varrho}$ and $\bar{\varrho}$ of the interaction $T \circledast S$. By courtesy, they are still O-forking and joined by an event $\bar{e} \leq e$. We can apply Lemma 5.37 and conclude that either $(\Pi_1 \lceil \bar{\varrho} \rceil^\sigma, \Pi_1 \lceil \bar{\varrho}' \rceil^\sigma)$ or $(\Pi_2 \lceil \bar{\varrho} \rceil^\tau, \Pi_2 \lceil \bar{\varrho} \rceil^\tau)$ are O-forking gccs that are joined (by either $\Pi_1 \bar{e}$ or $\Pi_2 \bar{e}$).
- ($\sharp$-*innocence*) Similar reasoning using Lemma 5.38. □

Define a CHO-strategy $\sigma : \mathcal{S} \rightharpoonup !A$ to be innocent when its erasure to $\mathrm{nCG}^{\cong}_{\circledcirc}$ $\sigma : S \rightharpoonup !A$ is innocent. Since innocence is stable under weak isomorphism, the previous result induces:

THEOREM 5.40. *Innocent strategies (up to weak isomorphism) form a cartesian closed subcategory* $\mathrm{CHO}_{\mathtt{inn}}$ *of* CHO.

PROOF. Parallel composition preserves innocence, and all the structural morphisms are based on copycat which is innocent (as it is a forest). Other constructions (eg. curryfication) do not affect the underlying event structure. Finally, innocence is trivially preserved by weak-isomorphism. □

We now have a notion of innocence and well-bracketing suited to our setting. In the next chapter, we will see prove that the conjunction of these conditions actually characterizes purely functional nondeterministic concurrent computations.

CHAPTER 6

Intensional full abstraction for ndPCF

In chapter 4, we have seen that our interpretations of **ndPCF** into CHO are adequate. In particular this means that if two terms have observationally equivalent denotations, they are observationally equivalent. This relies on the fact that all contexts can be interpreted inside CHO. The converse, however, is not true since not all semantic tests can be defined as **ndPCF** contexts. In Chapter 5, we introduced conditions to cut down on the behaviours definable in CHO. The goal of this chapter is to show that, up to may testing, those conditions are enough to prove intensional full abstraction: that interpretation preserves observational equivalence in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$, the subcategory of CHO consisting in innocent and well-bracketed strategies. This is achieved by leveraging fruitful consequences of innocence and well-bracketing, to describe and decompose strategies. However, we do not prove full abstraction for must equivalence as several arguments do not scale to must equivalence: in particular that finite tests are enough (Section 3.1). It is not clear to us at this point how to fix these problems.

Note that throughout this chapter, we will informally refer to "intensional full abstraction" as simply full abstraction.

Outline of the chapter. In Section 1, we construct *reduced forms* of innocent strategies, that induce a notion of *finite strategies*. This is crucial in order to state a theorem of *finite definability*: every finite strategy can be defined in the syntax (up to may testing). In Section 2, we provide an inductive decomposition of strategies at higher-order type, showing that innocence and well-bracketing ensure that higher-order behaviours are those of the λ-calculus (Theorem 6.23): this is the main result of the chapter. This allows us to reduce finite definability at every type, to finite definability at first-order types. In Section 3, we finally prove intensional full-abstraction of both interpretations *for may-testing*.

Contributions of the chapter. The full abstraction result in a deterministic setting (for PCF), via reduced forms and decomposition is joint work with Pierre Clairambault. This chapter extends this result to **ndPCF** with may equivalence.

1. Reduced form of innocent strategies

This first section extracts *reduced forms* from innocent strategies of CHO. These reduced forms contain no redundant information (and in particular no nontrivial symmetry) and generalize P-view trees of HO game semantics. In particular, one can see these reduced forms as some sort of "P-view dags" (at least in the deterministic case). As a result, innocent strategies have two representations in our setting: the usual one, as a strategy in CHO (the expanded form) or their reduced form. On the one hand, reduced forms are more compact and can be finite, but

cannot be easily composed. On the other hand, expanded forms are always infinite but can be easily composed. This dichotomy is reminiscent of the situation in HO game semantics: innocent strategies as sets of plays or P-view trees.

1.1. Reduced form of an innocent strategy. One important consequence of innocence is that, given two negative events sharing the same justifier and with the same label, their future do not interfere with each other, and are isomorphic due to symmetry. This means that Opponent cannot gain more knowledge about the strategy by playing twice the same move, differing only by their copy index. As a result, the behaviour of the strategy is completely captured by the fragment where Opponent always plays with copy index zero.

We fix an innocent $\sigma : \mathcal{S} \rightharpoonup\, !A \in \mathrm{CHO}(A)$ for the rest of the section.

DEFINITION 6.1. The reduced form of $\mathcal{S}$ is S_{rf}, the projection of S to events $s \in S$ such that all negative events below σs in $!A$ have copy index zero.

Note that S_{rf} is down-closed in S, meaning that $\mathscr{C}(S_{\mathrm{rf}}) \subseteq \mathscr{C}(S)$. What is the exact status of the reduced form of σ? Even though it can be mapped to $!A$ it is not a strategy as it fails receptivity by construction. However, we can regard it as a strategy on a subarena of $!A$. Define $!^+A$, the subarena of $!A$ comprising those index functions $\alpha : [a] \to \mathbb{N}$ that are zero on negative events of $[a]$. As a result σ restricts to a map of event structures $\sigma_{\mathrm{rf}} : S_{\mathrm{rf}} \rightharpoonup\, !^+A$.

LEMMA 6.2. *The map σ_{rf} defines an innocent strategy in* $\mathrm{nCG}^{\cong}_{\circledcirc}(!^+A)$.

PROOF. Straightforward. □

What about symmetry? The arena $!^+A$ inherits an isomorphism family from $!A$. The reduced form S_{rf} also inherits a symmetry from S, but it is trivial:

LEMMA 6.3. *If $\theta : x \cong y \in \tilde{S}$ is a symmetry between $x, y \in \mathscr{C}(S_{\mathrm{rf}})$ then $\theta = \mathrm{id}_x$.*

PROOF. By induction on x. Assume that $\mathrm{id}_x : x \cong x$ can be extended to θ by $(s, s') \in {S_{\mathrm{rf}}}^2$. If the pair is nonnegative, then by thin $s = s'$. Otherwise, we remark that $\sigma s = \sigma s'$ as they have same label, justifier and copy index (zero). Receptivity of σ implies $s = s'$. □

Hence, we can also regard σ_{rf} as a thin strategy on the tcg $!^+A$. The construction $\sigma \mapsto \sigma_{\mathrm{rf}}$ clearly preserves weak isomorphism. Define a **reduced form** on an arena A as an innocent strategy of $\mathrm{nCG}^{\cong}_{\circledcirc}(!^+A)$.

EXAMPLE 6.4. As an example of a reduced form, consider the strategy $\mathtt{if}_{\mathtt{par}}$ (or, rather its curryfication, playing on $!(\mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B})$) interpreting the if construct concurrently (Figure 5 of Chapter 4). Its reduced form is given in Figure 1. Even though the diagrams have the same shape, there are minor differences:

- Opponent indices can be omitted since they are always zero,
- Player indices do not need to encode earlier negative indices.

Our reduced forms are simpler (no Opponent indices, no complicated Player indices) and at the same time formal because they describe exactly the event structure – no implicit unfolding of copy indices is swept under the carpet. In this case, it is even *finite* as $\mathbb{B}$ is a finite datatype.

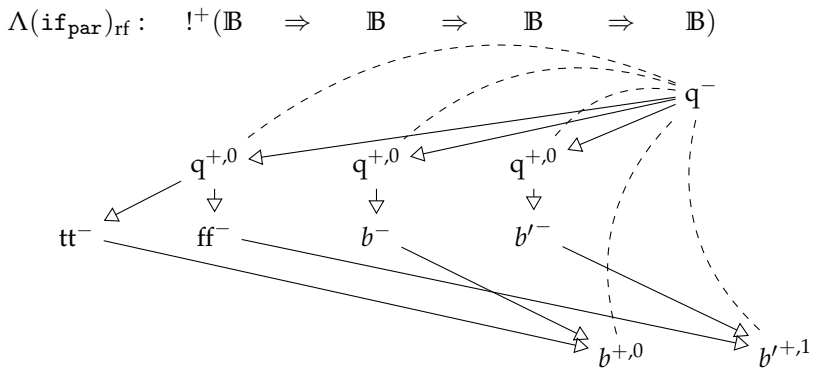


FIGURE 1. Reduced form for a parallel if

The main use of reduced forms is to give a notion of a finite strategies. A reduced form is **finite** when it contains a finite number of positive moves, and an innocent strategy of CHO is **finite** when its reduced form is. This is useful to prove finite definability (in particular of tests), by induction on the reduced form. Notice that not all finite reduced forms (in this sense) have a finite event structure: for instance, non constant reduced forms on $\mathbb{N} \Rightarrow \mathbb{N}$ are infinite by receptivity.

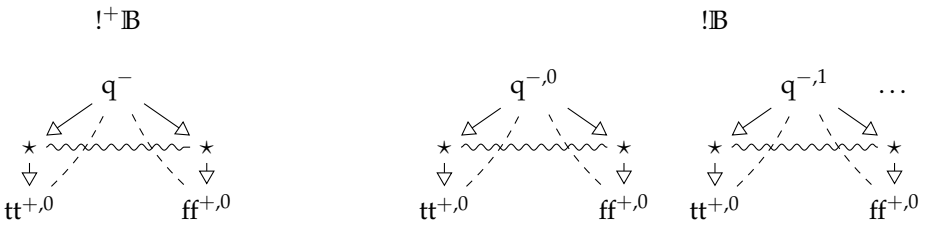
We now show how to go the other way: from a reduced form to a strategy. This expansion procedure has to automatically reconstruct the right copy indices of positive moves, depending on the indices of earlier negative moves.

1.2. Expansion of reduced forms. To define the expansion of a reduced form, we apply a construction similar to the expansion of arenas. Let $\sigma : S \rightharpoonup !^+A$ be a reduced form. We make the assumption that S is countable. We construct the event structure $!^-S$ as follows:

- *Events:* index functions $\alpha : [s] \to \mathbb{N}$ where $s \in S$ and $\alpha(s_0) = 0$ for any nonnegative $s_0 \leq s$. The event s is the label of α (written $\mathrm{lbl}(\alpha)$).
- *Causality:* inclusion of index functions
- *Conflict:* $\alpha : [s] \to \mathbb{N}$ and $\alpha' : [s'] \to \mathbb{N}$ when s is in conflict with s' in S and for all $s_0 \in [s] \cap [s']$, $\alpha(s_0) = \alpha'(s_0)$.

The convoluted definition of conflict means that two events are in conflict in the expansion if they are in conflict in the reduced form *and* do not come from different opponent branches.

EXAMPLE 6.5 (Conflict in the expansion). Consider the reduced form for `choice` on the left, and its expansion on the right:



The occurrence of $\mathrm{tt}^{+,0}$ above $\mathrm{q}^{-,0}$ and of $\mathrm{ff}^{+,0}$ above $\mathrm{q}^{-,0}$ project to conflicting events of the reduced form, yet they are not (and should not) be in conflict in the reduced form since their index for q^- differs.

As $!A$, $!^-S$ has a canonical isomorphism family, given by: $\theta : x \cong y \in \widetilde{!^-S}$ if and only if θ is an label-preserving order-isomorphism.

LEMMA 6.6. *$(!^-S, \widetilde{!^-S})$ is an event structure with symmetry.*

PROOF. Similar as proving that $\widetilde{!A}$ is an isomorphism family. □

To get a strategy, we need to explain how to map events from $!^-S$ to $!A$. This is not completely straightforward since we need to make sure that the resulting map is locally injective. As seen in Example 6.4, we only have constant copy indices in S, although we saw that in a typical strategy, the positive copy indices will depend on the copy indices of negative moves below it. To recover this dependency, we simply hash the negative history of the positive move along with its copy index:

$$\begin{array}{rcc} !^-\sigma : !^-S & \rightharpoonup & !A \\ (\alpha : [s_0] \to \mathbb{N}) & \mapsto & \begin{pmatrix} [\sigma s] \to \mathbb{N} \\ \sigma s^- \mapsto \alpha(s) \\ \sigma s^+ \mapsto \iota(\alpha|_{[s]}) \end{pmatrix} \end{array}$$

where, as before if $\sigma s = (a, \alpha)$ then $\mathrm{ind}(\sigma s) = \alpha(a)$ and $\iota : \bigcup_{X \in \mathscr{P}_f(S)} \mathbb{N}^X \to \mathbb{N}$ is an injective function from index functions to natural numbers (which exists since S is assumed countable). This copy index assignment is brutal and can be be optimized (for instance, copy indices below the justifier are not necessary).

LEMMA 6.7. *The map $!^-\sigma : !^-S \rightharpoonup !A$ defines an innocent* CHO*-strategy.*

PROOF. Clearly the symmetry on $!^-S$ is thin, and $!^-\sigma$ preserves symmetry.

Local injectivity. We proceed by induction on the product order S^2. Consider two consistent $\alpha : [s] \to \mathbb{N}, \alpha' : [s'] \to \mathbb{N}$ such that $!^-\sigma(\alpha) = !^-\sigma(\alpha')$. If s and s' are minimal, then they are negative and $\alpha : \{s\} \to \mathbb{N}$ is uniquely characterized by $(s, \alpha(s))$. Hence, $(\sigma s, \alpha(s)) = (\sigma s', \alpha(s'))$ implies by receptivity $s = s'$ and $\alpha = \alpha'$.

If s and s' are negative non-minimal, then we know that by induction $\mathrm{just}(\alpha) : [\mathrm{just}(s)] \to \mathbb{N}$ and $\mathrm{just}(\alpha) : [\mathrm{just}(s)] \to \mathbb{N}$ are equal. Since s and s' are negative, we have $[s] = [\mathrm{just}(s)] \cup \{s\}$, we just need to prove that $\alpha(s) = \alpha'(s)$:

$$\alpha(s) = !^-\sigma(\alpha)(s) = !^-\sigma(\alpha')(s) = \alpha'(s)$$

If s and s' are nonnegative, $!^-\sigma(\alpha) = !^-\sigma(\alpha')$ implies $\sharp(\alpha) = \sharp(\alpha')$ and $\alpha = \alpha'$.

Courtesy. Inherited from σ.

Strong receptivity. Assume $\theta : x \cong y \in \widetilde{!^-S}$ such that $!^-\sigma\theta$ extends by a negative pair (a_1, a_2). The justifiers of a_1 and a_2 belong to $!^-\sigma x$ and $!^-\sigma y$ respectively, let α_1 and α_2 be their pre-image in x and y. By receptivity of σ, there exist moves s_1 and s_2 in S whose justifiers are respectively $\mathrm{lbl}(\alpha_1)$ and $\mathrm{lbl}(\alpha_2)$ and labels in A those of a_1 and a_2. Define α_1' as the extension of α_1 with $\alpha_1'(s_1) = \mathrm{ind}\, a_1$ and α_2' as the extension of α_2 with $\alpha_2'(s_2) = \mathrm{ind}\, a_2$. We have that $\alpha_1', \alpha_2' \in !^-S$ and

$$\theta \cup \{(\alpha_1', \alpha_2')\} \in \widetilde{!^-S}.$$

Innocence. If $\alpha \rightarrow\!\!\!\!\triangleright\ \alpha'$ in $!^{-}S$, then we have $\mathrm{lbl}(\alpha) \rightarrow\!\!\!\!\triangleright\ \mathrm{lbl}(\alpha')$ in S. Hence $\rightarrow\!\!\!\!\triangleright$-innocence of $!^{-}S$ results from that of S. Consider two O-forking gccs ϱ, ϱ' of $!^{-}S$, forking at ϱ_i and racing at α_1 and α_2. We have two gccs $\mathrm{lbl}(\varrho_1)$ and $\mathrm{lbl}(\varrho_2)$. Since $\mathrm{lbl}(\alpha_1) \frown \mathrm{lbl}(\alpha_2)$, they are also racing. Moreover by definition of the conflict in $!^{-}S$, they must also be O-forking at $\mathrm{lbl}(\varrho_i)$. Indeed since ϱ_{i+1} and ϱ'_{i+1} are negative and distinct, with the same justifier and also the same copy index since $\alpha_1 \sharp \alpha_2$, we have that $\mathrm{lbl}(\varrho_{i+1}) \neq \mathrm{lbl}(\varrho'_{i+1})$. As a result, $\mathrm{lbl}(\varrho)$ and $\mathrm{lbl}(\varrho')$ are indeed racing at $\mathrm{lbl}(\alpha_1)$ and $\mathrm{lbl}(\alpha_2)$, contradicting the innocence of σ. □

Reducing the expansion of a reduced form leaves it invariant:

LEMMA 6.8. *For a reduced form $\sigma : S \rightharpoonup\ !^{+}A$, there is a weak isomorphism*

$$\sigma \approxeq (!^{-}\sigma)_{\mathrm{rf}}.$$

PROOF. Straightforward. □

Moreover, the expansion preserves weak isomorphism.

1.3. Equivalence between the two representations. We now prove that expansion and reduction are inverse of each other up to weak isomorphism. To prove this result, we need to study the normal configurations: those where Opponent uses a given Player move as a justifier at most once.

1.3.1. *Normal configurations.* Let us fix an innocent and countable strategy $\sigma : \mathcal{S} \rightharpoonup\ !A$. A configuration $x \in \mathscr{C}(S)$ is **normal** when there are no two negative events of x sharing the same justifier (where "sharing the same justifier" means that either they both do not have a justifier or both have the same justifier). Equivalently, x does not contain two O-forking gccs. Normal configurations cover what configurations are represented by the reduced form:

LEMMA 6.9. *For all normal configuration $x \in \mathscr{C}(S)$, there exists a unique normal configuration $y \in \mathscr{C}(S_{\mathrm{rf}})$ and a unique $\psi_x : x \cong y$ in $\tilde{S}$.*

PROOF. Replace the negative events with non-zero copy index by the ones that have copy index zero, inductively, using receptivity of σ. Uniqueness follows from Lemma 6.3. □

As a special case of this construction, since prime configurations are normal by $\rightarrow\!\!\!\!\triangleright$-innocence, we can map events of S to S_{rf}. This map is well-behaved:

LEMMA 6.10. *There exists a function $nf(\cdot) : S \to S_{\mathrm{rf}}$ such that*

(1) it preserves and reflects causal order,
(2) if $s \in S_{\mathrm{rf}}$, $nf(s) = s$,
(3) for all normal configuration $x \in \mathscr{C}(S)$ then $x \cong nf(x)$ in $\tilde{S}$,
(4) for a subset $x \subset S$ such that $nf(\cdot)$ is injective on x and $nf(x)$ is a normal configuration then x is a normal configuration,
(5) two events $s, s' \in S$ are in conflict if and only if $nf(s)$ and $nf(s')$ are in conflict, and $[s) \cup [s')$ is a normal configuration of S.

Note that this function is *not* a map of event structures (in particular it is not locally injective since all initial moves are collapsed). Condition (4) is reminiscent of the definition of the conflict in the expansion.

PROOF. For $s \in S$, by innocence, $[s]$ is normal hence there exists a *unique* (by Lemma 6.3) configuration $y \in \mathscr{C}(S_{\mathrm{rf}})$ with $[s] \cong y$ (Lemma 6.9). As a result, write $\mathrm{nf}(s)$ for the top-element of y, so that $y = [\mathrm{nf}(s)]$. Properties (2) and (3) follow by construction of $\mathrm{nf}(\cdot)$. (1) follows from $[s] \cong [\mathrm{nf}(s)]$.

(4) Because $\mathrm{nf}(\cdot)$ preserves and reflects causality, x is down-closed and has no distinct negative moves sharing the same justifier. We simply need to check that x is consistent. Assume there is a minimal conflict $s_1 \smile s_2$ in x. Write $y = [s_1] \cup [s_2) \subseteq x$: it is a normal configuration of S by $\sharp$-innocence. The symmetry $y \cong \mathrm{nf}(y)$ extends by $(s_2', \mathrm{nf}(s_2))$ for some $s_2' \in S$. By thin, we must have that $s_2' = s_2$, hence s_1 and s_2 are compatible: this contradicts the assumption.

(5) If s and s' are in conflict, then there is a minimal conflict $s_1 \smile s_1'$ with $s_1' \leq s$ and $s_1' \leq s'$. By $\sharp$-innocence, $[s_1) \cup [s_2)$ is a normal configuration hence mapped injectively by $\mathrm{nf}(\cdot)$. Hence so is the set $[s_1] \cup [s_2]$ which proves by (4) that its image in S_{rf} cannot be a configuration, ie. $\mathrm{nf}(s)$ and $\mathrm{nf}(s')$ cannot be compatible. Moreover, by $\sharp$-innocence, there cannot be any O-forking gcc in $[s] \cup [s']$.

Conversely, assume the right-hand side of the equivalence. If s and s' were compatible, then by assumption $[s] \cup [s']$ would be a *normal* configuration, and so would be $\mathrm{nf}([s] \cup [s'])$ implying that $\mathrm{nf}(s)$ and $\mathrm{nf}(s')$ are compatible. □

1.3.2. *Equivalence theorem.* The desired theorem follows from Lemma 6.10.

THEOREM 6.11. *For countable $\sigma \in \mathrm{CHO}_{\mathtt{inn}}(A)$, there is a weak isomorphism*

$$\sigma \approxeq !^{-}(\sigma_{\mathrm{rf}}).$$

PROOF. We define the isomorphism φ by $(s \in S) \mapsto (\alpha : [\mathrm{nf}(s)]^{-} \to \mathbb{N})$ where $\alpha(s_0^{-}) = \mathrm{ind}(\sigma s_0)$.

By Lemma 6.10.(5), φ preserves and reflects conflict: indeed conflict in S and $!^{-}S$ are similarly related to conflict in the reduced form. By property (1) of Lemma 6.10 it is a rigid map of event structures. An easy induction shows that if $\theta : x \cong y \in \tilde{S}$ then $\varphi\theta : \varphi x \cong \varphi y \in \widetilde{!^{-}S}$. Moreover, the converse is also true: if θ is a bijection $x \simeq y$ whose image by φ is in $\widetilde{!^{-}S}$ then it belongs to $\tilde{S}$.

Injectivity. Assume that $\varphi x = \varphi y$ for $x, y \in \mathscr{C}(S)$. The natural bijection $\theta_x : x \simeq \varphi x = \varphi y \simeq y$ is in $\tilde{S}$ as φ reflects symmetry. We show then by induction over x that this natural bijection is actually the identity, hence $x = y$ as desired. The base case is straightforward. Assume that $x' \subseteq x$ satisfies the inductive hypothesis $\theta_{x'} = \mathrm{id}_{x'}$ and can extend by $s \in x$. If s is nonnegative, we can conclude by thin. Otherwise, $\theta_x(s)$ is a negative event with the same justifier and label as s. Moreover, since $\varphi s = \varphi s'$ by definition of θ_x, they also have the same copy index, hence $s = s'$.

Surjectivity. By induction on $\alpha : [s_0] \to \mathbb{N} \in !^{-}S_{\mathrm{rf}}$ we build a $s \in S$ such that $\varphi s = \alpha$. If s_0 is negative and minimal, then this is a consequence of receptivity: S has a unique minimal negative event s such that $\mathrm{nf}(s) = s_0$ and $\mathrm{ind}(\sigma s) = \alpha(s_0)$.

If s_0 is negative and not minimal, by induction there exists $s \in S$ such that $\varphi s = \mathrm{just}(\alpha)$. By strong-receptivity, there exists an event $s' \in S$ whose justifier is s, and label and copy index is given by $\alpha(s_0)$. By construction $\varphi s' = \alpha$.

If s_0 is nonnegative, then applying the inductive hypothesis to all elements of $[\alpha)$ we get a set $x \subseteq S$ with $\varphi x = [\alpha)$. Since φ is rigid and reflects conflict, x is a configuration. Moreover, we have $x \cong [s_0)$ by construction. Since the right-hand

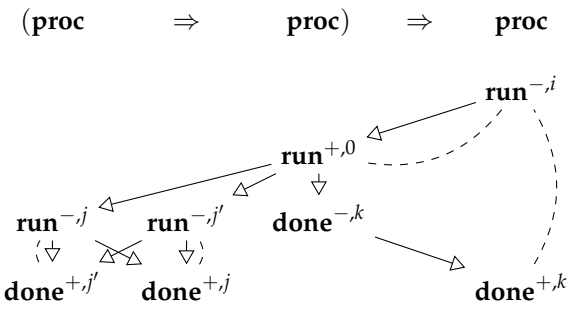


FIGURE 2. An ambiguous diagram for a non-innocent strategy

side extends by s_0, the left-hand side must extend by a unique $s \in S$. The negative moves in $[s)$ have index given by α, hence by definition of φ, $\varphi s = \alpha$. □

The following theorem has a very important consequence: to compare innocent strategies, there is no need to build an isomorphism at the level of event structures with symmetry, but simply at the level of event structures. As a result, the isomorphism family of an innocent strategy (up to weak isomorphism) can regarded as a simple property ("there exists an isomorphism family") since two choices of isomorphism families will result in weakly isomorphic CHO-strategies.

COROLLARY 6.12. *Let $\sigma : S \rightharpoonup !A$ and $\tau : T \rightharpoonup !A$ be countable strategies in $\mathrm{CHO}_{\mathtt{inn}}(A)$. The following statements are equivalent:*

(1) σ and τ are weakly isomorphic
(2) There exists an isomorphism of event structures *$\varphi : S \cong T$ with $\tau \circ \varphi \sim^{+} \sigma$*
(3) The reduced form σ_{rf} and τ_{rf} are weakly isomorphic.

PROOF. It is clear that $(1) \Rightarrow (2) \Rightarrow (3)$. We show that $(3) \Rightarrow (1)$.
By Theorem 6.11, it is enough to show that $!^{-}(\sigma_{\mathrm{rf}}) \cong !^{-}(\tau_{\mathrm{rf}})$. Since $\sigma_{\mathrm{rf}} \cong \tau_{\mathrm{rf}}$ by assumption, and expansion preserves weak isomorphism, we can conclude. □

As a result, the maps $\sigma \mapsto \sigma_{\mathrm{rf}}$ and $\sigma \mapsto !^{-}\sigma$ are well-defined on the equivalent classes up to weak isomorphism and inverse of each other.

1.4. Reduced form of non-innocent strategies. Before moving on to the rest of the argument of full abstraction, we would like to informally digress about the extension of reduced form outside the innocent case. In the innocent case, the previous section justifies the diagrams that we have been drawing with symbolic Opponent indices for innocent strategies. What our expansion does, is explaining how from the finite diagram of the strategy, the infinite mathematical object is generated. In particular, in the innocent case, our diagrams were non-ambiguous because of the equivalence between the representations: any other strategy that has this reduced form must be isomorphic to its expansion.

In this section, we present informally the problems that naturally arise when trying to generalize this picture to the non-innocent setting.

1.4.1. *Ambiguous expansion.* Remember the non-innocent strategy `bad` presented in the previous chapter (presented again in this chapter in Figure 2).

This strategy calls a function f with a special argument that returns as soon as f calls it twice. However, what is the expected behaviour when f calls its argument thrice? The question is legitimate, since in our world we can return *several times*. Hence, this diagram does not denote non-ambiguously a strategy. Indeed, to extend it to a strategy, we need at least to specify these missing causal links. Two possible ways are:

- When f evaluates its argument, it returns *once* for each previous invocation of itself, and returns every time for any further invocation.
- When f evaluates its argument, if f already invoked it before, it returns immediately, once. Otherwise it waits for another invocation and returns as soon as one occurs.

Note that the second option makes the strategy nondeterministic (there is a race). Also, those two choices give non-observational equivalent strategies (since f will be able to observe whether its argument might return twice).

As an example, we detail the interaction of the first choice against the term $\lambda x.(x_1 \parallel (x_2; x_3))$ (occurrences of x have been annotated to refer to them). How many answers at toplevel shall it yield? First, it depends on the implementation of $\parallel$ and ;. Assume that "$\parallel$" returns whenever its two arguments return – in other words, for every pair of answers from its arguments, it returns – whereas ";" spawns its second argument for each answer of its first, and returns all possible answers of all invocations of its second argument.

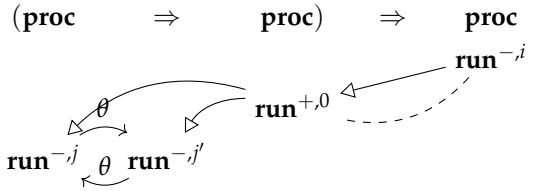
Operationally, this is what should happen during the interaction:

(1) First x_1 and x_2 are spawned in parallel so they both return.
(2) x_3 is then run which makes x_1 and x_2 return once more.
(3) the return of x_2 triggers a new invocation of x_3, and we go back to (2)

So x_1 is just enough to start a feedback loop between x_2 and x_3, hence we get infinitely many answers at toplevel. We spelled out this little example to demonstrate the complexity hidden behind the apparent simplicity (and finiteness) of the diagrams we have been drawing. Being able to avoid redundancy while still having a non-ambiguous representation of non-innocent strategies is a real challenge.

1.4.2. *Non-canonical symmetries.* A subtle consequence of Theorem 6.11 is that, even though once S is fixed, it could support several isomorphism families. However, they would all yield weakly isomorphic strategies (indeed any isomorphism on the event structures can be lifted to an isomorphism of event structures with symmetry by Theorem 6.11). Outside the non-innocent case, this is not the case anymore. Consider the following strategy description of Figure 3

Intuitively, given a function f, this strategy calls it and returns *twice* as soon as f called its argument twice. Note that this is not well-bracketed. Consider the symmetry $\theta : [s_1, s_2] \cong [s_1, s_2]$ permuting simply s_1 and s_2, where s_1, and s_2 are the two negative questions in the leftmost **proc**. The following diagram depicts θ:



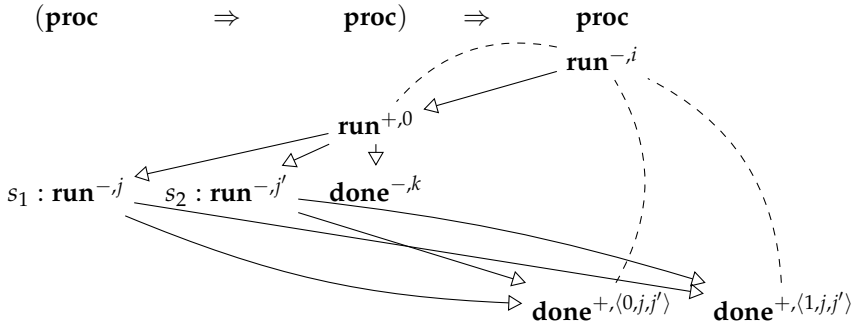


FIGURE 3. A strategy supporting several non-weakly isomorphic symmetries

Since $[s_1, s_2]$ can be extended by $s = \mathbf{done}^+_{\langle 0,j,j'\rangle}$, θ can be extended by (s, s') for some s'. However, in this case, there are two particular resolutions leading to two strategies: either $s' = s$ or $s' = \mathbf{done}^+_{\langle 1,j,j'\rangle}$. The two resulting strategies are **not** weakly isomorphic because of the symmetry, but are observationally equivalent. As a result, without innocence, symmetry is indeed a structure, and not a property of strategies.

1.4.3. *A syntax for non-innocent behaviours.* In the previous chapter, we have used IPA as a means to describe syntactically non-innocent strategies. However, IPA is not a good language for that purpose. First, it is well-bracketed so that we cannot express some important behaviours (such that returning multiple times). Moreover, references are a inadequate way of communication between different branches of the program, as they correlate causality and non-determinism (because of races). However, in our setting the two aspects are cleanly decomposed.

To end this prospective section, we would like to investigate some informal syntax to crudely account for the phenomena presented above.

Causality via signals. We should have a way to create causal links between parts of the program that are purely deterministic. To do so, we propose the usage of signal-like primitives that allows a signal to be fired in a certain part of the program and waited on in another. For instance, the syntax could be:

$$M ::= \ldots \mid \text{newsignal } s \text{ in } M \mid \text{wait } ([s_1, \ldots, s_n]) \mid \text{fire } ([s_1, \ldots, s_n])$$

Both waiting and firing are parametrized over a list of signals and produce a term of type **proc**. The idea is that wait $([s])$ returns every time the signal s is fired. Hence $M = (\text{fire } ([s]) \parallel \text{fire } ([s]) \parallel \text{wait } ([s]))$ would return twice to the top-level. The interest in firing and waiting on a list of signals is to be able to express subtle synchronization patterns. To illustrate informally the need for lists, consider the following two terms describing the two possible resolutions described previously.

$$M_1 = \lambda f.\, \text{newsignal } s, t \text{ in } f\ (\text{fire } ([s, t])) \parallel (\text{wait } ([s, t]); \texttt{skip})$$
$$M_2 = \lambda f.\, \text{newsignal } s, t \text{ in } f\ (\text{fire } ([s, t])) \parallel \text{wait } ([s, s]); (\texttt{skip} \parallel \texttt{skip})$$

Both terms call f in such a way that every time f evaluates its argument both signal are fired. Operationally, a list of signal occurrences is kept (a signal occurrence is a signal along with a natural number). When firing $[s, t]$, for instance, we add the signal occurrences (s, n) and (t, n) (where n is a fresh occurrence number)

to the list. Waiting on $[s,t]$ returns whenever we can find in the list an occurrence (s,k_1) and (t,k_2) with $k_1 \neq k_2$. Moreover if $s = t$, the order does not matter (so if the list contains $(s,1)$ and $(s,2)$, we only match once).

So, when f calls its argument twice we end up with the fired signals

$$[(s,1),(t,1),(s,2),(t,2)].$$

Then there is only one match for $[s,s]$ – namely $\{(s,1),(s,2)\}$ whereas there are two matches for $[s,t]$: $\{(s,1),(t,2)\}$ and $\{(s,2),(t,1)\}$.

In M_1, we still get two toplevel answers when f calls its argument twice, because there are two matches. Since the two events come from the same occurrences of fire $(\cdot)$, they will be symmetric in the strategy (corresponds to the $s \neq s'$ choice above). On the other hand, in M_2 there are two syntactically different occurrences of `skip` that are not symmetric (corresponds to the $s = s'$ choice above).

Nondeterminism via explicit races. Signals are purely deterministic. To add nondeterminism to this language, we use a construct to induce races between arbitrary parts of the program:

$$M := \ldots \mid \mathrm{race}(x,y).\, M$$

where in M, x and y are bound variables of type **proc**. The idea is to interpret race by a strategy similar to that depicted in Figure 2 (Chapter 5). When M evaluates x and y there is a race, only one of x and y will return (and allow the program to continue) but not both.

An implementation of parallel-or. As an example of expressivity, an implementation of parallel or in this setting would be:

$$\begin{aligned} \mathtt{por} = \lambda b.\lambda b'.\, &\mathrm{newsignal}\ s\ \mathrm{in}\ \mathrm{race}(x,y).\\ &(\mathtt{if}\ b\ (x;\mathrm{tt})\ (\mathrm{wait}\ ([s]);\mathrm{ff}))\\ \parallel\quad &(\mathtt{if}\ b'\ (y;\mathrm{tt})\ (\mathrm{fire}\ ([s]);\bot)) \end{aligned}$$

Signals are used to make sure that we only return ff when both evaluations have returned false, whereas races are used to prevent two answers at toplevel.

Having recovered a notion of finite strategies, we now explain how to decompose strategies at higher-order type inductively.

2. Higher-order decomposition of strategies

We now move to the main argument underlying the full-abstraction result: any *finite* innocent and well-bracketed strategy with a higher-order type can be defined by a λ-term that has access to all first-order strategies (Theorem 6.23). One can think of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ as the free dcpo-enriched CCC over its full subcategory of first-order types, though we will not aim at formalizing this intuition.

2.1. Properties of innocent and well-bracketed strategies.

To build the decomposition, we need a few more properties about strategies of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$.

LEMMA 6.13. *Let $\sigma : S \rightharpoonup A^{\perp} \parallel B$ be an innocent and well-bracketed strategy in $\mathrm{nCG}^{\cong}_{\circledcirc}$. For a question $q^+ \in S$, and a^+ an answer to $\mathrm{just}_e(q)$, q is answered in $[a]$.*

Note that since q is positive, it is not minimal, and $\mathrm{just_e(q)}$ is well-defined.

PROOF. Let a be an answer to $\mathrm{just_e(q)}$. By innocence of σ, the positive questions in $[a]$ must be well-answered, hence by well-bracketing, all questions of

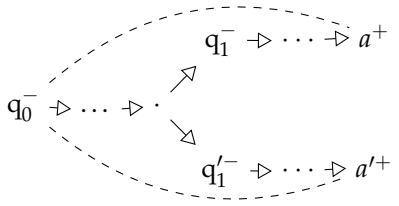
$[a]$ are well-answered. Since $\mathrm{just_e}(\mathrm{q})$ is answered in $[a]$, for $\mathrm{just_e}(\mathrm{q})$ to be well-answered in $[a]$, q must be answered there. □

2.1.1. *Well-bracketed gccs.* First, as hinted at in Section 1.1.3 (Chapter 5), in the presence of innocence and well-bracketing, threads are well-bracketed. We now make this claim formal. For a gcc $\varrho \in \mathrm{gcc}(S)$, the **pending question** at ϱ_i is the latest unanswered question in $\varrho_{0 \leq \_ \leq i}$. Call a gcc $\varrho \in \mathrm{gcc}(S)$ of a visible strategy $\sigma : \mathcal{S} \rightharpoonup\, !A$ **well-bracketed** when for every answer $\varrho_i \in \varrho$, the pending question in $\varrho_{\leq i}$ is $\mathrm{just}(\varrho_i)$.

LEMMA 6.14. *All gccs of a strategy in* $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ *are well-bracketed.*

PROOF. Let $\sigma : \mathcal{S} \rightharpoonup\, !A \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$, and consider ϱ a gcc of S which is not well-bracketed: there exists an answer $a \in \varrho$ that answers a question $\mathrm{q}_0 \in \varrho$ but the pending question $\mathrm{q}_1 \in \varrho$ at a is strictly after q_0. Hence we must have a positive (as Opponent always answers the previous question) and q_0 and q_1 negative, as well as the situation $\mathrm{q}_0 < \mathrm{q}_1 < a$. Write q_1' for a negative question which has the same justifier and label as q_1 but with an index that does not occur in $[a]$.

Consider the configuration $y = \{s \in [a] \mid s \not\geq \mathrm{q}_1\} \in \mathscr{C}(S)$. The identity on y extends to θ by $(\mathrm{q}_1, \mathrm{q}_1')$ in $\tilde{S}$ by $\sim$-receptivity. Since $y \cup \{\mathrm{q}_1\}$ extends to $[a]$, there exists $a' \in S$ and a further extension $\theta \subseteq \theta'$ such that $\theta' : [a] \cong [a']$, as follows:



The convoluted construction of θ' guarantees the following key property:

$$\theta' s \neq s \qquad \Leftrightarrow \qquad s \notin y \qquad \Leftrightarrow \qquad s \geq \mathrm{q}_1$$

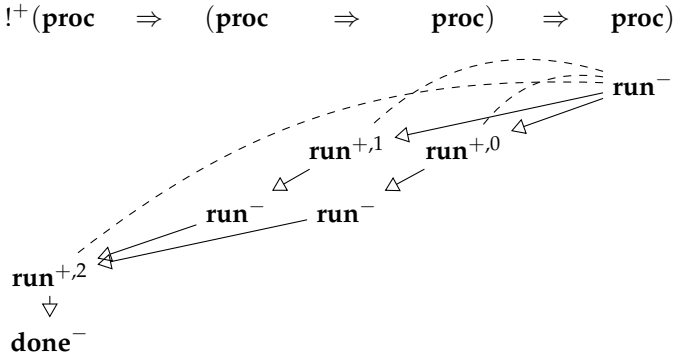
By innocence of σ, $[a, a']$ must be a configuration, in which q_0 is not well-answered because it has two answers: a and a'. By well-bracketing of σ, this implies that there exists a positive question q_2^+ in $[a, a']$ that is not well-answered. This means that there exist negative $a_2, m_2 \in [a, a']$ where a_2 is an answer to q_2 and m_2 is justified by q_2 but not an answered question.

By innocence, we cannot have a_2 and m_2 both below a or both below a'. Assume for instance $a_2 \leq a$ and $m_2 \leq a'$. First, since m_2 is negative, its only predecessor in S is q_2, its justifier. This means that $\mathrm{q}_1 \leq \mathrm{q}_2$ if and only if $\mathrm{q}_1 \leq m_2$.

If both hold, then $\mathrm{q}_1 \leq m_2 \leq a'$ and $\mathrm{q}_1' \leq a'$ contradicting innocence. If both do not hold, then $\theta' \mathrm{q}_2 = \mathrm{q}_2$ and $\theta' a_2 = a_2$. Applying θ' to $a_2 \leq a$ yields $a_2 \leq a'$, hence both m_2 and a_2 are below a' – a contradiction. □

2.1.2. *Complete threads.* We now prove that, up to observational equivalence, innocence and well-bracketing ensures that before a join (or a race), the two threads must be complete. In other words, it is not possible to merge two threads with

unanswered questions.[α] A prototypical example of such behaviours is the following strategy:



At the merge ($\mathbf{run}^{+,2}$), two negative $\mathbf{run}^{-}$ are unanswered. This strategy lives in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ but is not definable (up to weak isomorphism in **ndPCF**). However, there is no way to extend this strategy to contain an answer at toplevel: this would break well-bracketing. This strategy, even though not definable itself, is may equivalent to $\bot$, hence it is not a problem for intensional full abstraction. This intuition is captured by the following proposition:

PROPOSITION 6.15. *Let $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, B)$ be an innocent and well-bracketed strategy. The strategy σ_{cmpl} satisfies the following properties:*

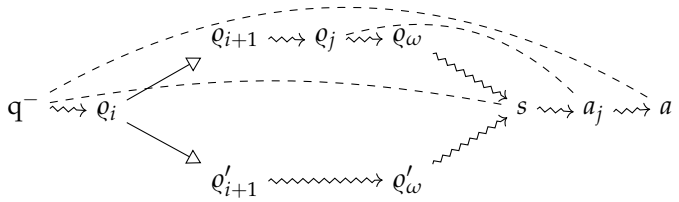
(1) For every forking gccs $\varrho, \varrho' \in \mathrm{gcc}(S_{\mathrm{cmpl}})$ whose divergence is $\varrho_i = \varrho'_i$ joined by a visible-bounded event, then the segments $\varrho_{>i}$ and $\varrho'_{>i}$ are complete.
(2) For every forking gccs $\varrho, \varrho' \in \mathrm{gcc}(S_{\mathrm{cmpl}})$ whose divergence is $\varrho_i = \varrho'_i$ that are racing at visible-bounded events, the segments $\varrho_{>i}$ and $\varrho'_{>i}$ are complete.

An event s is visible-bounded *when there exists $s_0 \in S_{\downarrow}$ with $s \leq s_0$.*

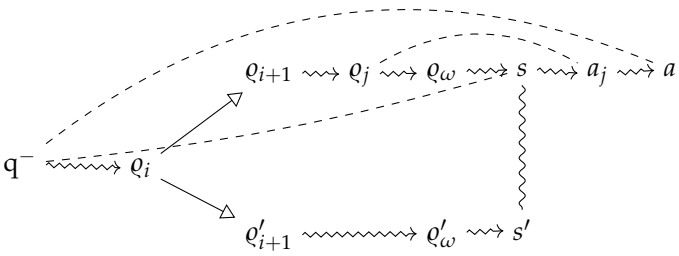
PROOF. *(1)* Consider a visible event $s \geq \varrho_\omega$ and $s \geq \varrho'_\omega$ which is visible. Without loss of generality, by courtesy, it can be chosen positive. Fix a completion x of $[s]$ (since $s \in S_{\mathrm{cmpl}}$). Assume that $\varrho_i = \varrho'_i$ is the divergence point of ϱ and ϱ'. Assume the segment $\varrho_{>i}$ is not complete for instance and let ϱ_j be a non-answered question in ϱ with $j > i$. Write $\mathrm{q} := \mathrm{just}_{\mathrm{e}}(s)$. By visibility, q must be before the fork ϱ_i. If s is an answer, this means that extensions of ϱ to s are not well-bracketed since their pending question is at least after ϱ_j. So s must be a (positive) question. Since x is complete, q must be answered by an event $a \in x$, and by Lemma 6.13, $s \leq a$. Since $\mathrm{q} \leq \varrho_i \leq \varrho_j \leq s \leq a$, there exist extensions of ϱ and ϱ' to $\bar{\varrho}$ and $\bar{\varrho}'$ such that $a \in \bar{\varrho} \cap \bar{\varrho}'$. Since $\bar{\varrho}$ is well-bracketed and $\mathrm{q} \leq \varrho_j$, it follows that ϱ_j is answered by some $a_j \geq s$ in $\bar{\varrho}$. This violates visibility, as $\varrho_j = \mathrm{just}_{\mathrm{e}}(a_j) \notin \bar{\varrho}'$ but $a_j \in \bar{\varrho}'$.

The proof is summed by the following picture (where $s \rightsquigarrow s'$ means $s \leq s'$):

[α] The reader familiar with **[CCW15]** may have recognized the conditions of well-bracketing used there. This notion – unlike the one presented here – was not stable under composition. We show here that, up to observational equivalence, we can recover those conditions that are key for finite definability.



(2) Assume ϱ and ϱ', positive $s \geq \varrho_\omega$ and $s' \geq \varrho'_\omega$ with $s \sharp s'$, and an unanswered question ϱ_j in the segment $\varrho_{>i}$. We follow a similar reasoning as for (1): because gccs are well-bracketed (Lemma 6.14), the event s must be a question. In a completion of $[s]$, $\mathrm{q} := \mathrm{just}_{\mathrm{e}}(s)$ has an answer a. By Lemma 6.13, $s \leq a$, and because gccs of σ_{cmpl} must be well-bracketed, there exists an answer a_j to ϱ_j between s and a, depicted as follows:



This time, $\sharp$-locality is violated: a_j is conflict with s'_2 but its justifier, ϱ_j, is concurrent to s'_2. □

We cannot apply directly this proposition to strategies of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$. Hence, in the decomposition procedure described in the next section, we consider strategies σ of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ satisfying the further assumptions (a) σ equal to its completion (ie. $\sigma_{\mathrm{cmpl}} = \sigma$) and (b) that all essential events are visible-bounded. Condition (b) erases behaviours up to must equivalence, but it does not matter here as we are only interested in full abstraction up to may testing. It is however not a fundamental problem: if one wanted a decomposition result robust for must equivalence, one can consider strategies satisfying conditions (1) and (2) of Proposition 6.15, as those conditions can be proven stable under composition.

In this chapter, we choose to instead show that every strategy of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ is may equivalent to a strategy satisfying (a) and (b):

LEMMA 6.16. *Let $\sigma : S \rightharpoonup !A$ be a well-bracketed CHO-strategy. There exists a strategy σ' such that $\sigma'_{\mathrm{cmpl}} = \sigma'$ and essential events of σ' are visible-bounded, which is may-equivalent to σ in* $\mathrm{CHO}_{\mathtt{wb}}$.

PROOF. Consider $\mathcal{S}'$, $\mathcal{S}$ projected to events $s \in \mathcal{S}$ such that

(1) s is observable (cf. Section 1.3 in Chapter 5),
(2) s lies below a visible event,
(3) if s is neutral, s is in minimal conflict with an observable and visible-bounded event.

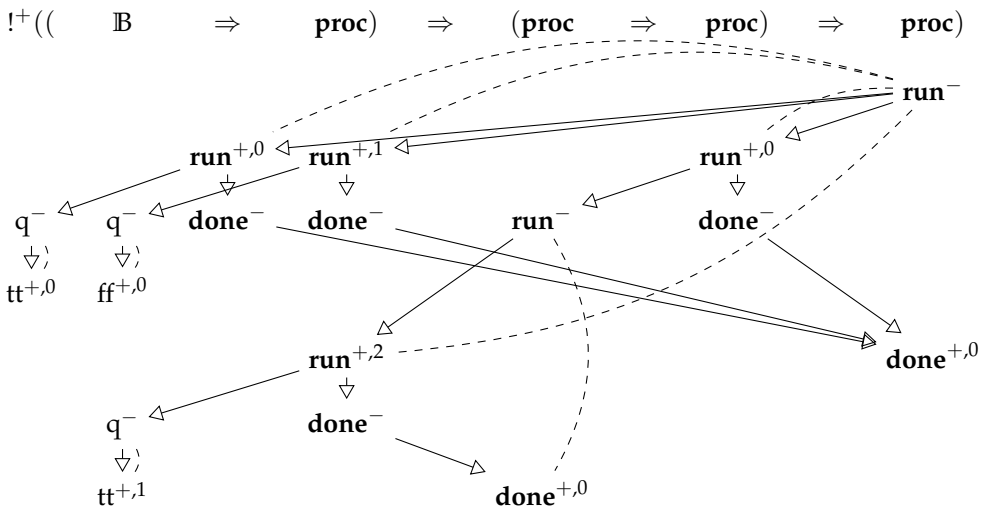


FIGURE 4. Reduced form of $[\![M]\!] = [\![\lambda f g.\, f_1\,\mathrm{tt} \parallel f_2\,\mathrm{ff} \parallel g\,(f\,\mathrm{tt})]\!]$

The restriction of σ yields a CHO-strategy $\sigma' : \mathcal{S}' \rightharpoonup A$ (point (3) makes sure it is an essential strategy). We show that σ' is may-equivalent to σ_{cmpl}, which in turn is may-equivalent to σ by Theorem 5.16.

Consider a well-bracketed test $\alpha \in \mathrm{CHO}(A, \mathbf{proc})$. Clearly, if $\alpha \circledcirc \sigma'$ may converge, so may $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$. Assume now that $\alpha \circledcirc \sigma_{\mathrm{cmpl}}$ may converge and consider a positive answer $a \in \alpha \circledcirc \sigma_{\mathrm{cmpl}}$. To show that the interaction $[a]_{\alpha \circledast \sigma_{\mathrm{cmpl}}}$ can be replayed in $\alpha \circledast \sigma'$, it is enough to show that $\Pi_1[a]_{\alpha \circledast \sigma_{\mathrm{cmpl}}} \in \mathscr{C}(S \parallel !\mathbf{proc})$ contains only visible bounded events. Assume it is not the case, and let $s \in \Pi_1[a]_{\alpha \circledast \sigma_{\mathrm{cmpl}}}$ be a maximal neutral event. Then, the corresponding $p \in [a]_{\alpha \circledast \sigma_{\mathrm{cmpl}}}$ such that $\Pi_1 p = s$ is maximal in $\alpha \circledast \sigma_{\mathrm{cmpl}}$ as α cannot put causal links from neutral events of S. This is absurd since it implies $p = a$. As a result $a \in \alpha \circledcirc \sigma'$ and $\alpha \circledcirc \sigma'$ may converge. □

2.2. Definition of the decomposition. Consider a higher-order PCF type $A = A_1 \Rightarrow \ldots \Rightarrow A_n \Rightarrow \mathbb{X}$ (where $\mathbb{X}$ is a ground type). Each A_i can be written $A_{i,1} \Rightarrow \ldots \Rightarrow A_{i,n_i} \Rightarrow \mathbb{X}_i$. The integer (possibly zero) n_i is called the **arity** of A_i. Write $\Gamma = A_1 \parallel \ldots \parallel A_n$ so that $A \cong \Gamma \Rightarrow \mathbb{X}$ (arena isomorphism).

Fix $\sigma : \mathcal{S} \rightharpoonup !A$ in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$. We decompose it in two parts:

- a first-order strategy σ_{flow} called the *flow*,
- a number of higher-order strategies σ_{q} called the *argument strategies*.

This decomposition can be illustrated at the syntax level. Consider the term

$$M = \lambda f g.\, f_1\,\mathrm{tt} \parallel f_2\,\mathrm{ff} \parallel g\,(f\,\mathrm{tt})$$

with $f : \mathbb{B} \Rightarrow \mathbf{proc}$ and $g : \mathbf{proc} \Rightarrow \mathbf{proc}$. Occurrences of f have been explicitly annotated. First, we look at the structure of toplevel function calls (called **primary questions** later), that forms the *flow* of M:

$$M_{\mathrm{flow}} = \lambda f_1 f_2 g.\, f_1 \parallel f_2 \parallel g$$

We split f into two arguments (one per occurrence) and we removed the arguments to the calls of f and g. The flow of M expects one argument for every toplevel call of M, representing their result. As a result M_{flow} has type **proc** $\Rightarrow$ **proc** $\Rightarrow$ **proc** $\Rightarrow$ **proc** – a first-order type.

Now, the calls needs arguments (one argument for each call in this example). The arguments are extracted from the term as follows:

$$M_{f_1} = \lambda fg.\,\mathrm{tt}$$
$$M_{f_2} = \lambda fg.\,\mathrm{ff}$$
$$M_g = \lambda fg.\,f\,\mathrm{tt}$$

Those will be called the **argument subterms**. Notice that they take the same arguments as M – arguments can have access to any variables M has access to. Finally, M can be recovered from its flow and its argument subterms as follows:

$$M =_{\beta\eta} \lambda f\lambda g.\, M_{\text{flow}}\,(M_{f_1}\,f\,g)\,(M_{f_2}\,f\,g)\,(M_g\,f\,g).$$

In this section, we perform a similar decomposition at the level of strategies. The steps of this construction will be illustrated on $[\![M]\!]$ depicted in Figure 4.

2.2.1. *The flow substrategy.* The flow substrategy captures the toplevel function calls made by σ. It is not concerned with what arguments are fed to these function calls. Define S_{flow} as the projection of S to those events $s \in S$ such that $\sigma_{\downarrow}[s] \subseteq\, !(\mathbb{X}_1 \Rightarrow \ldots \mathbb{X}_n \Rightarrow \mathbb{X})$ seen as a subarena of $!A$. Since S_{flow} is closed under symmetry, we get an event structure with symmetry $\mathcal{S}_{\text{flow}}$.

A **primary question** is a positive question $\mathrm{q}^+ \in S_{\text{flow}}$. Such questions are mapped to one of the $\mathbb{X}_i$ for some $i \in \mathbb{N}$ called its **index**. If q is a primary question of index i, we write $\mathbb{X}_{\mathrm{q}}$ for $\mathbb{X}_i$, A_{q} for A_i, $A_{\mathrm{q},j}$ for $A_{i,j}$ and Γ_{q} for $A_{\mathrm{q},1} \times \ldots \times A_{\mathrm{q},n_i}$, so that, in particular: $A_{\mathrm{q}} \cong \Gamma_{\mathrm{q}} \Rightarrow \mathbb{X}_{\mathrm{q}}$ (as arenas).

Write $\mathscr{Q}$ for the set of primary questions of σ and $\mathscr{Q}_i$ for those of index $i \in \mathbb{N}$. In our syntactic example we had $\mathscr{Q}_f = \{f_1, f_2\}$ and $\mathscr{Q}_g = \{g\}$. To construct the flow substrategy, one needs to slightly modify the restriction of σ as the flow strategy takes one argument *per* primary question, to have a map:

$$\sigma_{\text{flow}} : S_{\text{flow}} \rightharpoonup\ !((\prod_{\mathrm{q}\in\mathscr{Q}_i} \mathbb{X}_{\mathrm{q}}^{\perp} \parallel\ !\mathbb{X}).$$

Its domain is $S_{\text{flow}\downarrow}$. For a visible event $s \in S_{\text{flow}}$, define its image $\sigma_{\text{flow}}s \in\, !((\prod_{\mathrm{q}\in\mathscr{Q}_i} \mathbb{X}_{\mathrm{q}}^{\perp} \parallel\ !\mathbb{X})$ as follows, by induction on $s \in S_{\text{flow}}$:

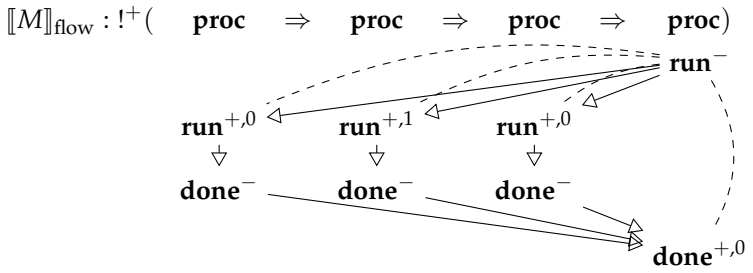
- If σs is mapped to $!\mathbb{X}$, then $\sigma_{\text{flow}}(s) = \sigma s$.
- If σs is a primary question q, then $\sigma_{\text{flow}}\mathrm{q}$ is mapped to the initial question of the component of $\prod_{\mathrm{q}\in\mathscr{Q}} \mathbb{X}_{\mathrm{q}}$ corresponding to q, and copy index given by $\mathrm{ind}(\sigma s)$.
- If σs is a (negative) answer to a primary question q, then $\sigma_{\text{flow}}s$ is the unique event in $!((\prod_{\mathrm{q}\in\mathscr{Q}} \mathbb{X}_{\mathrm{q}}^{\perp} \parallel\ !\mathbb{X})$ whose justifier is $\sigma_{\text{flow}}\mathrm{q}$, copy index and label that of σs.

LEMMA 6.17. *The map σ_{flow} defines an innocent and well-bracketed strategy.*

PROOF. Routine check. □

A remark that will be crucial later on is that σ_{flow} is linear in all its arguments: $\sigma_{\text{flow}}\mathrm{q}$ is the only positive move of $\sigma_{\text{flow}}(S_{\text{flow}}) \cap (!X_{\mathrm{q}})^{\perp}$.

Computing the flow for $[\![M]\!]$ gives a strategy whose reduced form is:

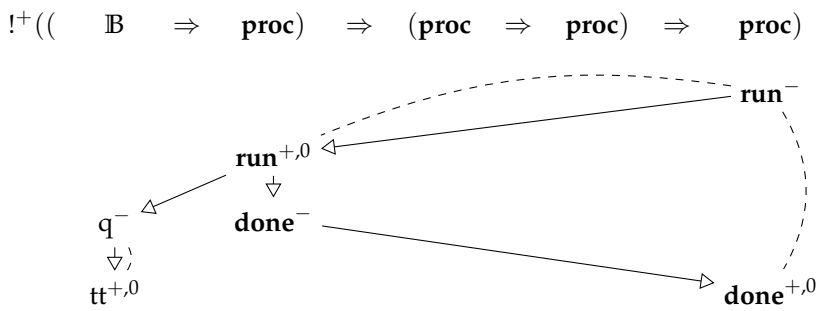


2.2.2. *Argument strategies.* We now move on to defining the argument strategies. Remember that a primary question $\mathrm{q} \in \mathscr{Q}_i$, corresponds to a call to the ith argument. As a result, σ is forced to provide n_i arguments to this call. The tuple formed by these arguments corresponds to a substrategy σ_{q} playing on $!(\Gamma \Rightarrow \Gamma_{\mathrm{q}})$ whose underlying event structure S_{q} is:

$$S_{\mathrm{q}} = S \downarrow \{s \in S \mid \exists s'^{\mathscr{Q}} \in [s], \mathrm{just}_{\mathrm{e}}(s') = \mathrm{q}\},$$

which contains the events of S lying above a question justified by q. Note that S_{q} does not contain any answer to q by courtesy.

On the example M, the argument strategy corresponding to the call of g gives a strategy whose reduced form is:



A key argument in the proof of the correctness of the decomposition is that it forms a partition of S:

LEMMA 6.18. *Assume that $\sigma = \sigma_{\mathrm{cmpl}}$ and that all events of S are visible-bounded. We have (as sets)*

$$S = S_{\mathrm{flow}} \cup \bigcup_q S_q$$

and moreover, this union is disjoint, and no conflict of S is spanning over two components of this union.

PROOF. It is easy to see that an event not in the flow must be greater than a question justified by a primary question, hence the equality of sets.

Disjoint decomposition. If $s \in S_{\mathrm{flow}} \cap S_{\mathrm{q}}$ for some q, then there is a contradiction between $\sigma[s]$ living in $!(\mathbb{X}_1 \Rightarrow \ldots \Rightarrow \mathbb{X})$ and being above a question justified by q.

If $s \in S_{\mathrm{q}} \cap S_{\mathrm{q}'}$ for $\mathrm{q} \neq \mathrm{q}'$, then either $\mathrm{q} < \mathrm{q}'$, and there must be a (negative) answer to q in between, forcing $[s]$ to contain an answer and a question justified

by q, contradicting $\rightarrowtriangle$-innocence. Or, q and q′ are concurrent, and s being visible-bounded, contradicts property (1) of Proposition 6.15.

No conflict. Consider a minimal conflict between $s_1 \in S_{\text{flow}}$ and $s_2 \in S_{\text{q}}$ for some $\text{q} \in \mathcal{Q}$. If $s_1 < \text{q}$ then $s_1 < s_2$, absurd. If $\text{q} < s_1$, then s_1 depends on a negative answer to q, and the conflict $s_1 \smile s_2$ violates $\sharp$-innocence. Finally, if q and s_1 are concurrent, $s_1 \smile s_2$ contradicts property (2) of Proposition 6.15. □

As the set S_{q} is closed under symmetry, $\mathcal{S}_{\text{q}} = \mathcal{S} \downarrow S_{\text{q}}$ is an event structure with symmetry that can be mapped to the game $!\Gamma^{\perp} \parallel !\Gamma_{\text{q}}$. Remember that σ can be uncurried to $\Lambda^{-1}(\sigma) : S \rightharpoonup\ !\Gamma^{\perp} \parallel !\mathbb{X}$. We define σ_{q} on $(S_{\text{q}})_{\downarrow}$:

- Every event s that is hereditarily justified by q (meaning $\sigma s > \sigma \text{q}$ – hence in particular is projected to $A_{\text{q},j}$ for some j) gets redirected to the return type occurrence of $A_{\text{q},j}$ (this is compatible with polarities – since the occurrence of $A_{i,j}$ in A is twice to the left of an implication.)
- Otherwise, we remark that $\sigma s \notin !\mathbb{X}$. Indeed, in this case, it would be a (positive) answer, and gccs of s would not be well-bracketed since they would contain an answer to the initial question before q were answered, contradicting Lemma 6.14. This means that $\Lambda^{-1}(\sigma)(s)$ lives in $!\Gamma$ and we let: $\sigma_{\text{q}} s = \Lambda^{-1}(\sigma)(s)$.

LEMMA 6.19. *The map σ_q is an innocent and well-bracketed strategy from Γ to Γ_q.*

PROOF. Routine check. □

2.3. The decomposition theorem. We now prove that from the strategy σ_{flow} and the family $(\sigma_{\text{q}})_{\text{q}}$, we can recover σ up to weak isomorphism. First, let us introduce some notations. Define

$$\epsilon_{\text{q}} = \Gamma \xrightarrow{\langle \sigma_{\text{q}}, \pi_i \rangle} \Gamma_{\text{q}} \times (\Gamma_{\text{q}} \Rightarrow \mathbb{X}_{\text{q}}) \xrightarrow{\text{ev}} \mathbb{X}_{\text{q}} \quad \text{(composition in } \text{CHO}_{\texttt{inn},\texttt{wb}}\text{)}.$$

This strategy is called the *evaluator* for q.

We can then state our decomposition result:

THEOREM 6.20. *Assume $\sigma = \sigma_{\text{cmpl}}$ and all events of S are visible-bounded. There is a weak isomorphism*

$$\sigma \approxeq \sigma_{\text{flow}} \circledcirc \langle \epsilon_q \mid q \in \mathcal{Q} \rangle$$

The proof of this theorem is detailed in the rest of the section.

2.3.1. *Characterisation of the evaluators.* To establish our result, we first give a simpler characterisation of the evaluators $\epsilon_{\text{q}} \in \text{CHO}_{\texttt{inn},\texttt{wb}}(\Gamma, \mathbb{X}_q)$. By Lemma 6.12, to compare innocent strategies, symmetries can be ignored, and by Lemma 2.50 it is equivalent to compare their configuration domains. Finally, since our strategies are visible, interaction is deadlock-free so securedness is always verified.

As a result, configurations of the interaction $T \circledast S$ corresponds to pairs that we will write $y \circledast x$ of a configuration $x \in \mathscr{C}(S)$ and $y \in \mathscr{C}(T)$ such that they synchronize in the middle. This condition restricts the shape of x and y and will be called the *synchronization condition*. Moreover, a configuration $y \circledast x$ of $T \circledast S$ is a configuration $T \circledcirc S$ when its maximal elements are visible or essential. This further restricts the shape of x and y and the corresponding condition will be called the *hiding condition*.

Finally, given a configuration x on the expansion of a composite arena (eg. $x \in \mathscr{C}(!((A \parallel B) \Rightarrow C))$) we will write $x^A \in \mathscr{C}(!A^{\perp})$ its projection to the A component.

In particular, any configuration $x \in \mathscr{C}(!(A \parr B))$ decomposes faithfully in $x^A \in \mathscr{C}(!A)$ and $x^B \in \mathscr{C}(!B)$ via the isomorphism $m_{A,B} : !(A \parr B) \cong !A \parr !B$. However, the decomposition of $x \in \mathscr{C}(!(A \Rightarrow B)$ into $x^A \in \mathscr{C}(!A^\perp)$ and $x^B \in \mathscr{C}(!B)$ obtained via $\chi_{A,B} : !(A \Rightarrow B) \to !A^\perp \parr !B$ is not injective: the causal links from x^B to x^A are lost, and more precisely, for an event in x^A, the minimal event in x^B it depends on. However, it is when x has a unique minimal event, as then there is no ambiguity.

First, we characterize the post-composition by the evaluation:

LEMMA 6.21. *Let $\Gamma = C \parr (A \Rightarrow \mathbb{X})$ be a context and let $\sigma : \mathcal{S} \rightharpoonup !\Gamma^\perp \parr !A$ be in* $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\Gamma, A)$. *Consider:*

$$\epsilon = \Gamma \xrightarrow{\langle \sigma, \pi_2 \rangle} A \parr (A \Rightarrow \mathbb{X}) \xrightarrow{\mathrm{ev}} \mathbb{X}_q.$$

Configurations of ϵ with a unique minimal event are order-isomorphic to

$$\{(x \in \mathscr{C}(S), z_1 \parr z_2 \in \mathscr{C}(\mathbb{CC}_{!\mathbb{X}})) \mid x \neq \emptyset \Rightarrow z_1 \neq \emptyset\}$$

We only characterize the configurations of ϵ that have a unique minimal event as it makes the characterization simpler. We will be only using this lemma on configurations of ϵ_q that come from the interaction with σ_{flow}, which is linear, so those configurations will indeed have a unique minimal event.

PROOF. Remember that the evaluation strategy is obtained from copycat: a configuration of ev is a configuration of $z_1 \parr z_2 \in \mathbb{CC}_{!(A \Rightarrow \mathbb{X})}$. It is mapped to

$$z_1^A \parr z_2 \parr z_1^{\mathbb{X}} \in \mathscr{C}(!A^\perp \parr !(A \Rightarrow \mathbb{X})^\perp \parr !\mathbb{X}).$$

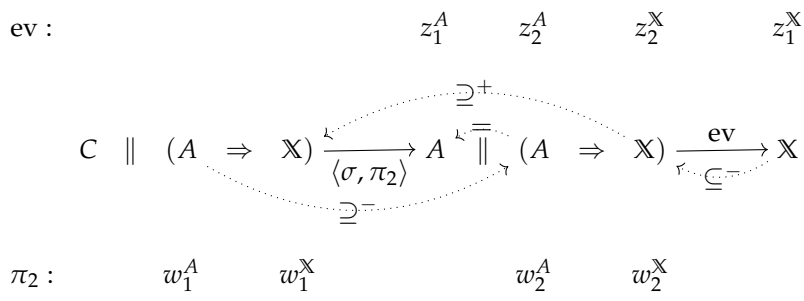
Similarly, configurations of π_2 are also given by configuration $w_1 \parr w_2 \in \mathbb{CC}_{!(A \Rightarrow \mathbb{X})}$ (mapped to itself since $!(A \Rightarrow \mathbb{X})$ is a sub-game of $!\Gamma$). Overall, a configuration of ϵ is a tuple

$$\underbrace{(z_1 \parr z_2)}_{\in \mathscr{C}(\mathbb{CC}_{!(A \Rightarrow \mathbb{X})})} \circledast \underbrace{(x_S}_{x \in \mathscr{C}(S)} \parr \underbrace{(w_1 \parr w_2))}_{\in \mathscr{C}(\mathbb{CC}_{!(A \Rightarrow \mathbb{X})})}.$$

The synchronization conditions give:

- for σ and ev on $!A$: $\sigma^A x = z_1^A$
- for π_2 and ev on $!(A \Rightarrow \mathbb{X})$: $w_2 = z_2$.

The hiding conditions are given by the following diagram summing up the interaction between the different copycats involved:



An arc connects two dual occurrences of a game connected by copycat. It flows from the negative one (ie. those where the minimal moves are negative)

occurrence to the positive one. The symbol on the arrow sums up the relations between the two configurations, in a configuration of the interaction whose maximal events are visible or essential. As a result, we get the following hiding conditions:

- $z_1^A = z_2^A = w_2^A$.
- $z_1^{\mathbb{X}} \supseteq^- z_2^{\mathbb{X}} = w_2^{\mathbb{X}} \subseteq^+ w_1^{\mathbb{X}}$, hence $z_1^{\mathbb{X}} \sqsubseteq w_1^{\mathbb{X}}$, in particular $w_2^{\mathbb{X}} = w_1^{\mathbb{X}} \cap z_1^{\mathbb{X}}$.
- $w_1^A \supseteq^- w_2^A$.

At this point, we are tempted to define the desired isomorphism as:

$$(x, w_1 \parallel w_2, z_1 \parallel z_2) \mapsto (x, w_1^{\mathbb{X}} \parallel z_1^{\mathbb{X}})$$

as it is well-defined. Moreover, from $(x, w_1^{\mathbb{X}}, z_1^{\mathbb{X}})$, we can recover $z_1^A = z_2^A$ via $\sigma^A x$. As a result, since z_2 has a unique minimal element (and so does z_1), both z_1 and z_2 are determined by their projections on $!A$ and $!\mathbb{X}$. Similarly, $w_2^{\mathbb{X}}$ and w_2^A are determined, so w_2 is also determined. The missing determination is w_1^A, as we only know that $\sigma^A x \subseteq^- w_1^A$. However, by receptivity, x extends uniquely to $\mathrm{ext}(x, w_1^A)$ so that $\sigma^A(\mathrm{ext}(x, w_1 A) = y$, hence the following map is the desired order-isomorphism:

$$(x, w_1 \parallel w_2, z_1 \parallel z_2) \mapsto (\mathrm{ext}(x, w_1^A), w_1^{\mathbb{X}} \parallel z_1^{\mathbb{X}})$$

Injectivity comes from the synchronization and hiding conditions, surjectivity and order preservation and reflection is routine. □

2.3.2. *Putting it together.* We can now prove Theorem 6.20:

THEOREM 6.20. *Assume $\sigma = \sigma_{\mathrm{cmpl}}$ and all events of S are visible-bounded. There is a weak isomorphism*

$$\sigma \approxeq \sigma_{\mathrm{flow}} \circledcirc \langle \epsilon_q \mid q \in \mathscr{Q} \rangle$$

PROOF. Write τ for the right-hand-side. A configuration of τ corresponds to a pair $[(x_{\mathrm{q}})_{\mathrm{q} \in \mathscr{Q}}, x_f]$ with $x_{\mathrm{q}} \in \mathscr{C}(\epsilon_{\mathrm{q}})$ and $x_f \in \mathscr{C}(S_{\mathrm{flow}})$. By Lemma 6.21, since x_f contains one minimal question of each $\mathbb{X}_{\mathrm{q}}$ at most, each x_{q} further decomposes in $(y_{\mathrm{q}}, w_{\mathrm{q}} \parallel z_{\mathrm{q}}) \in S_{\mathrm{q}} \times \mathscr{C}(CC_{!\mathbb{X}_{\mathrm{q}}})$ with $y_{\mathrm{q}} \neq \emptyset \Rightarrow w_{\mathrm{q}} \neq \emptyset$. The synchronization conditions gives us that $\sigma_{\mathrm{flow}}(x_f)^{\mathbb{X}_{\mathrm{q}}} = w_{\mathrm{q}}$, which shows that w_{q} is redundant (determined by x_f). Moreover, the hiding condition give us $w_{\mathrm{q}} = z_{\mathrm{q}}$.

Define $x = x_f \cup \bigcup_{\mathrm{q}} y_{\mathrm{q}} \in \mathscr{C}(S)$, consistent by Lemma 6.18, and down-closed because if $y_{\mathrm{q}} \neq \emptyset$, $\mathrm{q} \in x_f$. By the previous remarks, the mapping $[(x_f, (x_{\mathrm{q}}))] \mapsto x$ is injective since other components are determined from only x.

Conversely, by Lemma 6.18, we can write $x \in \mathscr{C}(S)$ uniquely in the form

$$x = x_f \cup \bigcup_{\mathrm{q} \in \mathscr{Q}} x_{\mathrm{q}}$$

Because the flow is linear, we know that each x_{q} has a unique minimal event. Hence $(x_{\mathrm{q}}, \sigma_{\mathrm{flow}}^{\mathbb{X}_{\mathrm{q}}} x_f \parallel \sigma_{\mathrm{flow}}^{\mathbb{X}_{\mathrm{q}}} x_f)$ corresponds to a configuration e_{q} of ϵ_{q} by Lemma 6.21. Then the configuration $[x_f, (e_{\mathrm{q}})_{\mathrm{q}}]$ is a configuration of τ as desired. □

If we have a finite reduced form, we can keep applying this decomposition to argument substrategies of σ. For instance in M, the argument strategy for the call of g (which corresponds to $\lambda fg.\, f\,\mathrm{tt}$) can still be further decomposed in a flow $(\lambda f.\, f)$ with one primary question, and a single argument substrategy $\lambda fg.\,\mathrm{tt}$. The decomposition stops there since $\lambda fg.\,\mathrm{tt}$ does not contain any primary questions.

What we are left with is a λ-term M that depends on finitely many first order strategies representing flows of substrategies. This intuition gives the following lemma:

THEOREM 6.23 (Higher-order definability). *Let $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ be a finite strategy. There exists a PCF term $x_1 : B_1, \dots, x_n : B_n \vdash M : A$ where the B_i are first-order types, and strategies $\tau_i \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(B_i)$ such that*

$$\sigma \simeq_{may} [\![M]\!] \circledcirc \langle \tau_1, \dots, \tau_n \rangle$$

Remark that M can be assumed without fixpoint: it is in fact a pure λ-term.

PROOF. First, if σ does not satisfy the conditions of Theorem 6.20, we replace it by the strategy obtained from Lemma 6.16 which is may equivalent to σ.

This is a corollary of Theorem 6.20: the only difficulty is to show that the decomposition does not actually need product types (unavailable in PCF), as all considered strategies can be curried. We proceed by induction on the number of positive moves in the reduced form of σ. If there are none, then $M = \bot$ (with $n = 0$) satisfies the required condition. Otherwise, we know that

$$\sigma \cong \sigma_{\mathrm{flow}} \circledcirc \langle \epsilon_{\mathrm{q}} \mid \mathrm{q} \in \mathscr{Q} \rangle.$$

Remember that

$$\epsilon_{\mathrm{q}} = \Gamma \xrightarrow{\langle \sigma_{\mathrm{q}}, \pi_i \rangle} \Gamma_{\mathrm{q}} \times (\Gamma_{\mathrm{q}} \Rightarrow \mathbb{X}_{\mathrm{q}}) \xrightarrow{\mathrm{ev}} \mathbb{X}_{\mathrm{q}}.$$

Define $\sigma_{\mathrm{q},j} = \pi_j \circledcirc \sigma_{\mathrm{q},j} : \Gamma \Rightarrow A_j$ for $1 \leq j \leq n_i$. Its currying $\Lambda(\sigma_{\mathrm{q},j})$ is a strategy on $A_1 \Rightarrow \dots \Rightarrow A_p \Rightarrow \mathbb{X}_j$. Since its reduced form is strictly smaller than that of σ (it is a subset of that of σ_{q}), it contains less positive moves, and by induction corresponds to a term $M_{\mathrm{q},j}(\vec{x}_{\mathrm{q},j})$ and first-order strategies $\vec{\tau}_{\mathrm{q},j}$ such that $[\![M_{\mathrm{q},j}]\!] \circledcirc \langle \vec{\tau}_{\mathrm{q},j} \rangle$. We assume that $\vec{x}_{\mathrm{q},j}$ are disjoint for different j or different q. Then,

$$E_{\mathrm{q}} := \lambda a_1 \dots a_p.\, a_i\, (M_{\mathrm{q},1}\, a_1 \dots\, a_p) \dots\, (M_{\mathrm{q},n_i}\, a_1 \dots\, a_p)$$

is such that $\epsilon_{\mathrm{q}} \cong [\![E_{\mathrm{q}}]\!] \circledcirc \langle \vec{x}_{\mathrm{q},j} \rangle_j$. Finally, we recover, for $\mathscr{Q} = \{\mathrm{q}_1, \dots, \mathrm{q}_n\}$:

$$\sigma \cong [\![y\, E_{\mathrm{q}_1} \dots\, E_{\mathrm{q}_n}]\!] \circledcirc \langle \sigma_{\mathrm{flow}}, (\vec{\tau_{\mathrm{q},j}})_{\mathrm{q},j} \rangle$$

where y is a free variable chosen outside the $(\vec{x}_{\mathrm{q},j})_{\mathrm{q},j}$ (put first in the context). □

3. Intensional full-abstraction for ndPCF

In this section, we compare the observational equivalences on the semantic side to those on the syntactic side. More precisely, we show that two terms are may equivalent if and only if any of their interpretations (the parallel or the sequential one) are may equivalent.

To do so, we leverage the results obtained in the previous sections. The notion of reduced form allows us to define a notion of *finite* strategies. Moreover, we have shown how to use this reduced form to neatly decompose a strategy into a first-order strategy and "smaller" higher-order strategy. This shows that with innocent and well-bracketed strategies, there are no higher-order patterns that are not definable in λ-calculus, all the expressive power resides in the first-order strategies.

The outline of this section is as follows:

(1) In Section 3.1, we prove that finite tests have enough discriminating power to characterise may equivalence.

(2) In Section 3.2, we show how to represent the may equivalence behaviour of strategies on ground and first-order types, by relations.
(3) In Section 3.3, we show how to define first-order *finite* strategies in **ndPCF**.
(4) In Section 3.4, we finally prove full abstraction by assembling the pieces of the puzzle.

Note that, the hard part about extending the full abstraction result to must equivalence is point (1).

3.0.1. *Observational equivalence.* In the category $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$, we have a coarser observational notion of equivalence since there are fewer tests, less behaviours can be explored:

DEFINITION 6.24. Two strategies in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ are **may-equivalent** if they may pass the same tests in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, \mathbf{proc})$.

A test $\alpha \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, \mathbf{proc})$ is finite when α as a strategy is finite (its reduced form contains a finite number of positive move). Naturally, observational equivalence is a congruence:

LEMMA 6.25. *Let $\sigma, \sigma' \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, B)$ such that $\Lambda(\sigma) \simeq \Lambda(\sigma')$ and $\tau, \tau' \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(B, C)$ such that $\Lambda(\tau) \simeq \Lambda(\tau')$. Then $\Lambda(\tau \circledcirc \sigma)$ and $\Lambda(\tau' \circledcirc \sigma')$ are observationally equivalent.*

PROOF. Consequence of the CCC structure of CHO. □

3.1. Finite tests. We now proceed to show that finite tests are enough to distinguish strategies up to may equivalence.

Let $\sigma : \mathcal{S} \rightharpoonup\, !A \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$. Write $\sigma_{\mathrm{rf}} : S_{\mathrm{rf}} \rightharpoonup\, !^+A$ and $\mathrm{nf}_\sigma : \mathcal{S} \to S_{\mathrm{rf}}$. Given a subset $X \subseteq S_{\mathrm{rf}}$ which is maximal for $\subseteq^-$, define $\mathcal{S}_X = \mathcal{S} \downarrow \mathrm{nf}_\sigma^{-1}(X)$ (well-defined because $\mathrm{nf}_\sigma^{-1}(X)$ is closed under symmetry).

LEMMA 6.26. *The restriction of σ to $\mathcal{S}_X$ defines a strategy $\sigma_X \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$.*

PROOF. Straightforward. (Receptivity comes from X being $\subseteq^-$-maximal.) □

A strategy of the form σ_X is called an **approximation of** σ. If X has a finite number of positive moves, σ_X is a a **finite approximation** of σ.

LEMMA 6.27. *Let $\alpha \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, \mathbb{B})$ and $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$. We have:*

- *If $X \subseteq Y$ are subsets of the reduced form of α, then $\mathscr{C}(\alpha_X \circledcirc \sigma) \subseteq \mathscr{C}(\alpha_Y \circledcirc \sigma)$.*
- *If $\mathfrak{X}$ is a set of subsets of the reduced form of α then,*

$$\mathscr{C}(\alpha_{\bigcup \mathfrak{X}} \circledcirc \sigma) = \bigcup_{X \in \mathfrak{X}} \mathscr{C}(\alpha_X \circledcirc \sigma).$$

PROOF. Straightforward. □

LEMMA 6.28. *Let $\sigma, \tau \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ such that there exists a $\alpha \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, \mathbb{B})$ such that $\alpha \circledcirc \sigma$ may converge but not $\alpha \circledcirc \tau$. Then, there exists a finite such α.*

PROOF. From Lemma 6.27, it is easy to see that σ may pass α if and only if σ may pass a finite approximation of α, which entails the result. □

3.2. First-order extensional behaviour. In this section, we characterize the may-equivalence behaviour at first-order types, as the equality of some induced functions. This proves that, up to observational equivalence, our terms have a functional behaviour. (Or rather, since we are in a nondeterministic world, a relational behaviour).

3.2.1. *Ground types.* We start with ground types. For a ground type $\mathbb{X}$, we write $\mathcal{X}$ for the set of values of that type (eg. $\mathcal{B} = \{\mathsf{tt}, \mathsf{ff}\}$) and $\downarrow \mathbb{X}$ for $\mathscr{P}(\mathcal{X})$. We first show that $\downarrow \mathbb{X}$ captures the quotient of $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X})$ up to may-equivalence:

LEMMA 6.29. *There exist a bijection* $\downarrow\colon \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X})/\simeq_{may} \cong \downarrow \mathbb{X}$.

PROOF. Given $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X})$, define $\downarrow \sigma$ as the set of $v \in \mathcal{X}$ such that there is a move labelled v in σ. First, we show it is well-defined. Assume that $\sigma \simeq_{\mathrm{may}} \tau$. If $\downarrow \sigma = \emptyset$, then $\neg(\sigma \Downarrow_{\mathrm{may}})$. As a result $\neg(\tau \Downarrow_{\mathrm{may}})$ and $\downarrow \tau = \emptyset$. Otherwise, assume for instance $\mathbb{X} = \mathbb{B}$. If $\mathsf{tt} \in\downarrow \sigma$, then σ may pass the test $[\![\lambda b.\, \mathtt{if}\, b\, \mathsf{tt}\, \bot]\!]$. Hence τ may pass this test and $\mathsf{tt} \in\downarrow \sigma$. Similarly for ff hence $\downarrow \sigma =\downarrow \tau$.

The map $\downarrow$ is easily checked to be surjective, so we tackle injectivity now. Let $\sigma, \tau \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X})$ such that $\downarrow \sigma =\downarrow \tau$ and $\alpha \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X}, \mathbb{B})$. Assume that $\alpha \circledcirc \sigma$ may converge: there is a configuration x of $\alpha \circledcirc \sigma$ that contains a positive move. If $\Pi_2[x] \in \mathscr{C}(\alpha)$ does not have any positive questions, then α is constant and $\alpha \circledcirc \tau$ may converge as well. Otherwise, there might be several positive questions, each of them having an answer (played by σ), yielding finally an answer. Since $\downarrow \sigma =\downarrow \tau$, both σ and τ can play the same answers and it is easy to extract from x a configuration y of $\alpha \circledcirc \tau$ which has a positive event, hence proving that τ may pass α as well, hence $\downarrow$ is indeed a bijection. □

3.2.2. *First-order strategies as functions.* Given $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X}_1 \times \ldots \mathbb{X}_n, \mathbb{Y})$ we define

$$\begin{aligned} \downarrow \sigma : \mathscr{P}_f(\mathcal{X}_1) \times \ldots \times \mathscr{P}_f(\mathcal{X}_n) &\to \mathscr{P}(\mathcal{Y}) \\ (V_1, \ldots, V_n) &\mapsto \downarrow (\sigma \circledcirc \langle \uparrow V_1, \ldots, \uparrow V_n \rangle) \end{aligned}$$

where $\uparrow$ is the inverse of $\downarrow$ on ground types. We need to remember the action on finite sets of values (representing nondeterministic values), and not a single value. For instance $\lambda x.\, \bot$ and $\lambda x.\, \mathtt{if}\, x\, (\mathtt{if}\, x \,\bot\, \mathsf{tt})\, \bot$ have the same behaviour on deterministic inputs, but the latter may converge on $x = \mathtt{choice}$.

In this section, we show the following theorem:

THEOREM 6.30. *Two strategies* $\sigma, \tau \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X}_1 \times \ldots \times \mathbb{X}_n, \mathbb{Y})$ *are may-equivalent if and only if* $\downarrow \sigma =\downarrow \tau$.

The rest of the section is dedicated to the proof. We provide the proof in a simplified case, where $n = 1$ as to convey the ideas of the proof clearly.

3.2.3. *Question outcome.* To prove this result, we look at the reduced form of tests. By Lemma 6.28, we know we can only consider finite tests α. Consider a finite $\alpha \in \mathrm{CHO}(\mathbb{X} \Rightarrow \mathbb{Y}, \mathbb{B})$. We write T for the event structure underlying α, and T_{rf} for the reduced form of T. If q is a question answered by a, we write $[q, a] := \{s \mid q \leq s \leq a\}$ for the segment. At order below two, because of innocence and well-bracketing, this only be contains questions and their answers.

The main difficulty is to show that if σ and τ have the same extensional behaviour ($\downarrow \sigma =\downarrow \tau$), then the initial questions of $\alpha \circledast \sigma$ and $\alpha \circledast \tau$ have the same answers. For that, we define a notion of "correct answer" with respect to a function $\mathscr{P}_f(\mathcal{X}) \to \mathscr{P}(\mathcal{Y})$ in T_{rf}, which are the answers due to occur in a pullback against any σ verifying $\downarrow \sigma = f$.

Consider a function $f : \mathscr{P}_f(\mathcal{X}) \to \mathscr{P}(\mathcal{Y})$. The reverse order $>_{T_{\mathrm{rf}}}$ is well-founded since T_{rf} contains finitely many positive moves, and causal chains $s_0 \rightarrowtriangle$

$s_1 \rightarrowtriangle \dots$ are alternating. By induction on $>_{T_{\mathrm{rf}}}$, we define for each question $\mathrm{q} \in T_{\mathrm{rf}}$, a set $\mathscr{O}_f(\mathrm{q})$ of potential answers as follows:

- If q is positive, then q is the initial question of $\mathbb{Y}$, and by receptivity there exists a (unique) negative question q_1 which is a successor of q. We let
$$\mathscr{O}_f(\mathrm{q}) = \{a \mid \mathrm{just}(a) = \mathrm{q} \;\&\; \exists A_1 \subseteq \mathscr{O}_f(\mathrm{q}_1), \mathrm{lbl}\, a \in f(\mathrm{lbl}(A_1))\}$$
This is well-defined because $\mathscr{O}_f(\mathrm{q}_1)$ is finite as q_1 is negative.
- If q is negative, we let
$$\mathscr{O}_f(\mathrm{q}) = \{a \in T \mid \mathrm{just}(a) = \mathrm{q} \;\&\; \text{for } q < a' < a,\, a' \in \mathscr{O}_f(\mathrm{just}(a'))\}.$$
This is well defined: if $a' < a$, well-bracketing entails (via Lemma 6.14) that $\mathrm{just}(a') > \mathrm{q}$. Note that this set is always finite because T_{rf} has finitely many positive moves.

This function $\mathscr{O}(\cdot)$ assigns to each question of T_{rf} its outcome in any interaction against a strategy σ implementing f. An answer a of T is **correct for** f when $a \in \mathscr{O}_f(\mathrm{just}(a))$. Positive question (asked to Opponent) should be answered according to f (first case), where as negative questions (asked by Opponent) should only be answered by answers depending previous correct answers.

To prove that this intention is actually met, we define the actual outcome of a question in the interaction as follows. Given $\sigma \in \mathrm{CHO}(\mathbb{X}, \mathbb{Y})$ and a question $\overline{\mathrm{q}} \in \alpha \circledast \sigma$ we define $\overline{\mathscr{O}_\sigma}(\overline{\mathrm{q}}) = \{\mathrm{lbl}(\bar{a}) \mid \mathrm{just}(\bar{a}) = \overline{\mathrm{q}}\}$ which is the set of answers to $\bar{\mathrm{q}}$. We write $\mathfrak{p} = \mathrm{nf} \circ \Pi_2 : \alpha \circledast \sigma \to \alpha_{\mathrm{rf}}$ to project an event of the interaction onto the reduced form of α. Throughout the proof, we denote events of a pullback with a bar: $\bar{\mathrm{q}}, \bar{a}, \dots$, while their projections by $\mathfrak{p}$ is simply written $\mathrm{q}, a, \dots$

LEMMA 6.31. *For $\sigma \in \mathrm{CHO}(\mathbb{X}, \mathbb{Y})$ and all $\overline{q} \in \alpha \circledast \sigma$, we have*
$$\mathfrak{p}(\overline{\mathscr{O}_\sigma}(\overline{q})) = \mathscr{O}_{\downarrow\sigma}(\mathfrak{p}\overline{q}).$$

This lemma states that for any question of an interaction with σ, its answer within this interaction coincide with the answers within T_{rf} correct with respect to $\downarrow\sigma$.

We proceed to prove Lemma 6.31 by induction on $>_T$.

3.2.4. *Lemma 6.31 – Negative case.* Let $\overline{\mathrm{q}^-}$ be a negative question of $\alpha \circledast \sigma$ and $\mathrm{q} := \mathfrak{p}\overline{\mathrm{q}}$ its projection.

Let $\bar{a} \in \overline{\mathscr{O}_\sigma}(\overline{\mathrm{q}})$, and write $a := \mathfrak{p}\bar{a}$. Consider an answer a' to q' in $[q, a]$, and $\bar{a}'$ and $\bar{\mathrm{q}}'$ their corresponding events in $[a]$. By definition $\bar{a}' \in \mathscr{O}_\sigma(\bar{\mathrm{q}}')$ hence by induction $a' \in \mathscr{O}_{\downarrow\sigma}(\mathrm{q}')$. This proves that $a = \mathfrak{p}\bar{a} \in \overline{\mathscr{O}_{\downarrow\sigma}}(\mathrm{q})$.

Conversely, let $a \in \mathscr{O}_{\downarrow\sigma}(\mathrm{q})$. If $\mathrm{q} \rightarrowtriangle a$ in T_{rf}, then it is easy to build a corresponding $\bar{a} \in \alpha \circledast \sigma$ such that
$$a = \mathfrak{p}\bar{a} \in \mathscr{O}_\sigma(\bar{\mathrm{q}})$$

Otherwise, $[q, a]$ contains only positive questions and their negative answers (apart from q and a). By induction each question can be played by τ, and each answer τ expects will be played by σ, since by induction those answers are correct. This means all the answers a depends on will be played, and finally τ is able to play a resulting in the desired event $\bar{a} \in \alpha \circledast \sigma$ with $\mathfrak{p}\bar{a} = a$. Hence, $a \in \mathfrak{p}(\mathscr{O}_\sigma(\bar{\mathrm{q}}))$.

3.2.5. *Lemma 6.31 – Positive case.* Let $\overline{\mathsf{q}} \in \alpha \circledast \sigma$ be a question such that $\mathsf{q} := \mathfrak{p}\overline{\mathsf{q}}$ is positive. We write q_1 for the unique question justified by q in T_{rf}.

Let $\bar{a} \in \overline{\mathscr{O}_\sigma}(\overline{\mathsf{q}})$. To show that $a := \mathfrak{p}\bar{a} \in \mathscr{O}_{\downarrow\sigma}(\mathsf{q})$, we need to show that a is a correct answer. Write q_1 be the unique question justified by q in T_{rf}. In σ, the segment $\Pi_1[\mathsf{q}, a]$ looks like:

$$\Pi_1\mathsf{q}^- \twoheadrightarrow \Pi_1\mathsf{q}_1^0 \twoheadrightarrow a_1^- \twoheadrightarrow \ldots \twoheadrightarrow \Pi_1\mathsf{q}_1^n \twoheadrightarrow \Pi_1 a_k^- \twoheadrightarrow a.$$

This very rigid structure is forced by well-bracketing and innocence. Write $A_1 = \{a' \in [a] \mid \mathfrak{p}\,\mathrm{just}(a') = \mathsf{q}_1\}$ (which is $\{a_1, \ldots, a_k\}$ in the example) for the set of answers provided to σ by α. By definition of $\downarrow\sigma$ we must have $\mathrm{lbl}(a) \in f(\mathrm{lbl}(A_1))$, as desired.

Conversely, let $a \in \mathscr{O}_{\downarrow\sigma}(\mathsf{q})$. We know that there exists $A_1 \subseteq \mathscr{O}_{\downarrow\sigma}(\mathsf{q}_1)$ such that $\mathrm{lbl}(a) \in\downarrow \sigma(\mathrm{lbl}(A_1))$. By induction, we know that for each $a_i \in A_i$, there exists $\bar{a}_i \in \alpha \circledast \sigma$ with $\mathfrak{p}\bar{a}_i = a_i$. They must all be consistent, since they are justified by different copies of q_1. The configuration $x = [\bar{a}_1, \ldots, \bar{a}_n]$ must extend by $\bar{a} \in \alpha \circledast \sigma$ such that $\bar{a} = a$ since $\mathrm{lbl}(a) \in\downarrow \sigma(\mathrm{lbl}(A_1))$. Hence, $a \in \mathfrak{p}(\mathscr{O}_\sigma(\mathsf{q}))$.

3.2.6. *Wrapping up.* We can now prove Theorem 6.30:

PROOF. (Of Theorem 6.30) If $\sigma \simeq_{\mathrm{may}} \tau$, since observational equivalence is a congruence (Lemma 6.25), we have: $\downarrow\sigma =\downarrow \tau$.

Conversely, assume that $\downarrow\sigma =\downarrow \tau$ and assume there exists a test $\alpha \in \mathrm{CHO}(\mathbb{X} \Rightarrow \mathbb{Y}, \mathbb{B})$ such that $\alpha \circledcirc \sigma$ may converge but not $\alpha \circledcirc \tau$. By Lemma 6.28, we can assume that α is finite. Assume $\overline{\mathsf{q}_\sigma}$ and $\overline{\mathsf{q}_\tau}$ are initial questions of $\alpha \circledast \sigma$ and $\alpha \circledast \tau$ respectively, both projecting to the same initial question q of T_{rf} via $\mathfrak{p}$. By Lemma 6.31:

$$\mathfrak{p}(\overline{\mathscr{O}_\sigma}(\overline{\mathsf{q}_\sigma})) = \mathscr{O}_{\downarrow\sigma}(\mathsf{q}) = \mathscr{O}_{\downarrow\tau}(\mathsf{q}) = \mathfrak{p}(\overline{\mathscr{O}_\tau}(\overline{\mathsf{q}_\tau})).$$

However, the left-hand side must be non-empty since $\alpha \circledcirc \sigma$ may converge and the right-hand side must be empty because $\alpha \circledcirc \sigma$ is not may-convergent. □

3.3. First-order finite definability. To conclude full abstraction, we need to prove finite definability. The only ingredient missing is definability on first-order types. In this section, we consider $A = \mathbb{X}_1 \Rightarrow \ldots \mathbb{X}_n \Rightarrow \mathbb{X}$ a first-order type.

3.3.1. *Image of a strategy.* We will not prove finite definability up to may equivalence (which is all we need for full-abstraction) but up to a stronger equivalence. Given a $\sigma : S \rightharpoonup\, !A \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$, its **image** is the set

$$\mathfrak{i}(\sigma) := \{\sigma[a] \mid a^+ \in S_{\mathrm{rf}}\} \subseteq \mathscr{C}(!A)$$

Unsurprisingly, having the same image is stronger than may equivalence:

LEMMA 6.32. *Let $\sigma, \tau \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$. If $\mathfrak{i}(\sigma) = \mathfrak{i}(\tau)$ then $\sigma \simeq_{may} \tau$.*

PROOF. Assume $\mathfrak{i}(\sigma) = \mathfrak{i}(\tau)$. To show $\sigma \simeq_{\mathrm{may}} \tau$, it is enough to show $\downarrow\sigma =\downarrow \tau$ by Theorem 6.30. By symmetry, we only show $\downarrow\sigma \subseteq\downarrow \tau$ (point-wise inclusion).

Let $(V_1, \ldots, V_n) \in \mathscr{P}_f(\mathbb{X}_1) \times \ldots \times \mathscr{P}_f(\mathbb{X}_n)$, and $y \in\downarrow \sigma(V_1, \ldots, V_n)$. By definition this means that there exists a positive answer a with label y in $\sigma \circledcirc \langle\uparrow V_1, \ldots, \uparrow V_n\rangle$. As a result, $[a]$ corresponds to a pair of a configuration $[a_S] \in \mathscr{C}(\sigma)$ and $w \in \mathscr{C}(\langle\uparrow V_1, \ldots, \uparrow V_n\rangle)$ matching on $\mathbb{X}_1 \times \ldots \times \mathbb{X}_n$.

Since σ is innocent, $[a_S]$ is normal, and by Lemma 6.9, we can assume that a_S is an element of the reduced form. As a result $\sigma[a_S] \in \mathfrak{i}(\sigma) = \mathfrak{i}(\tau)$, and there exists $a_T \in T_{\mathrm{rf}}$ such that $\sigma[a_T] = \tau[a_T]$. As a result, $[a_T]$ and w are matching on

$\mathbb{X}_1 \times \ldots \times \mathbb{X}_n$. By the deadlock-free lemma (Theorem 5.35), they correspond to a configuration $[a'] \in \tau \circledcirc \langle \uparrow V_1, \ldots, \uparrow V_n \rangle$ witnessing that $y \in \downarrow \tau(x_1, \ldots, x_n)$. □

3.3.2. *Deterministic case.* We first prove that finite *deterministic* strategies are defined with a method along the lines of **[CCW15]**. First, we introduce a bit of vocabulary. Given a deterministic reduced form $\sigma : S \to !^+A$ where A is the arena of a first-order type, a **minimal action** is a successor of the unique initial move of S. Minimal actions are always positive (there are no essential events since σ is deterministic). If $s \in S$ is a minimal action which is an answer, then S_{rf} has only two elements.

If s is a question, then it is slightly more complicated. Pick s' a negative move justified by s. Because A is first-order, s' is an answer. In that case, we define

$$S/(s,s') = S \downarrow \{s'' \in S \mid s'' \notin [s'] \wedge \neg(s'' \sharp s') \wedge (s'' > s \Rightarrow s'' > s')\}.$$

The third condition ensures that if s'' causally depends on the question s, then it is in the branch of s'. Indeed, in $S/(s,s')$ we only want to keep those events that are either in a call concurrent to s or in the branch given by the answer s'. Again, we get a reduced form $\sigma/(s,s')$ whose expansion is in $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ if that of σ is.

LEMMA 6.33. *Let $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X}_1 \times \ldots \mathbb{X}_n, \mathbb{X})$ be a deterministic finite first-order strategy. There exists a term M of **ndPCF** such that $\mathrm{i}(\llbracket M \rrbracket) = \mathrm{i}(\sigma)$.*

PROOF. Write S for the event structure of the reduced form of σ. We proceed by induction over S. If S has no minimal actions, then $\sigma \cong \llbracket \bot \rrbracket$. Otherwise, pick a minimal action $s \in S$. There are three cases:

- It is a positive answer with label $v \in \mathbb{X}$. As discussed before, $S_{\mathrm{rf}} \cong \mathrm{q} \rightarrowtriangle v$ in that case, hence: $\sigma \cong\!\!\!\!\approx \llbracket \lambda x_1 \ldots x_n.\, v \rrbracket$.
- It is a positive question. Write $s_1, \ldots, s_n$ for the set of negative answers to s (that exist by receptivity) such that a positive answer lies above them. Since S is finite, we know that there must be a finite number of such s_i, even if the datatype is infinite. Then we get terms $M_1, \ldots, M_n$ by induction on $\sigma/(s,s_1), \ldots, \sigma/(s,s_n)$. Then it is easy to define a case construct in **ndPCF** that will test for the value of the corresponding argument and plug the corresponding term. Since we have a finite number of cases, we get a finite term M which satisfies the required condition. □

3.3.3. *First-order finite definability – general case.* For a nondeterministic strategy, the situation is a bit more complicated. Because we have few axioms structuring conflicts in first-order strategies, an inductive construction as in the deterministic case is harder to define. However, we know that the observational behaviour of such strategies are very simple: they are simply relations.

LEMMA 6.34. *Let $\sigma : S \rightharpoonup A \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$, and $x \in \mathrm{i}(\sigma)$. There exists a* deterministic *strategy $\sigma_x \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ such that $\mathrm{i}(\sigma_x) = \{x\}$.*

PROOF. Let $\sigma[a] \in \mathrm{i}(\sigma)$ where a is a positive answer of the reduced form. Consider the restriction $\sigma : [a] \rightharpoonup !^+A$. It is almost a valid reduced form: it fails receptivity. Write y for the maximal negative extension of $[a]$ (well-defined as there are no minimal conflicts involving negative events). It is routine to check that $\sigma : y \rightharpoonup !^+A$ is an innocent and well-bracketed strategy of $\mathrm{nCG}^{\cong}_{\circledcirc}$. Its expansion is the desired strategy. □

Write $\mathtt{sum} = [\![\lambda x \lambda y.\, \mathtt{if}\ \mathtt{choice}\, x\, y]\!] \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(\mathbb{X} \times \mathbb{X}, \mathbb{X})$ for any ground type $\mathbb{X}$, and write $\sigma + \tau = \mathtt{sum} \circledcirc \langle \sigma, \tau \rangle$.

LEMMA 6.35. *Let $\sigma, \tau \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$. We have:*

$$\mathfrak{i}(\sigma + \tau) = \mathfrak{i}(\sigma) \cup \mathfrak{i}(\tau)$$

PROOF. This is an easy calculation, since configurations of the reduced form of $\sigma + \tau$ containing at least a positive move are in one-to-one correspondence with those of the disjoint union of those of σ and τ. □

THEOREM 6.36. *Let $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ be a finite strategy. There exists a term $\vdash M : A$ such that $[\![M]\!] \simeq_{may} \sigma$.*

PROOF. Since σ is finite, $\mathfrak{i}(\sigma)$ is also a finite set. By Lemmata 6.33 and 6.34, for each $x \in \mathfrak{i}(\sigma)$, there exists $\vdash M_x : A$ with $\mathfrak{i}([\![M_x]\!]) \simeq_{\mathrm{may}} \mathfrak{i}(\sigma_x) = \{x\}$. Then writing

$$M = \mathtt{if}\ \mathtt{choice}\, M_{x_1} \ldots (\mathtt{if}\ \mathtt{choice}\, M_{x_{n-1}}\, M_{x_n})$$

we have

$$\mathfrak{i}(M) = \mathfrak{i}(\sigma_{x_1}) \cup \ldots \cup \mathfrak{i}(\sigma_{x_n}) = \mathfrak{i}(\sigma),$$

by Lemma 6.35. □

3.4. Wrapping up. We can now conclude finite definability:

LEMMA 6.37. *Let $\sigma \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A)$ be a finite strategy. Then there exists a term M such that $[\![M]\!]$ is observationally equivalent to σ.*

PROOF. First, using Theorem 6.23, we know that:

$$\sigma \simeq_{\mathrm{may}} [\![M]\!] \circledcirc \langle \tau_1, \ldots, \tau_n \rangle$$

where $x_1 : B_1, \ldots, x_n : B_n \vdash M : A$ and $\tau_i \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(B_i)$ are strategies on a first-order type. By Theorem 6.36, there exist terms $\vdash N_i : B_i$ with $[\![N_i]\!] \simeq_{\mathrm{may}} \tau_i$. Then it is easy to see that

$$\sigma \simeq_{\mathrm{may}} [\![M[N_i/x_i]]\!]. \quad \square$$

And, finally, we conclude intensional full abstraction as well:

THEOREM 6.38 (Intensional full abstraction). *Both interpretations of **ndPCF** inside $\mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}$ are intensionally fully abstract.*

PROOF. Let M, N be closed terms of type A.

Assume $[\![M]\!]$ and $[\![N]\!]$ are observationally equivalent, and let a context $\mathcal{C}[\,]$ distinguishing them. Since $[\![\mathcal{C}[M]]\!] \cong [\![\mathcal{C}[\,]]\!] \circledcirc [\![M]\!]$ and $[\![\mathcal{C}[N]]\!] \cong [\![\mathcal{C}[\,]]\!] \circledcirc [\![N]\!]$, it follows that $[\![\mathcal{C}[\,]]\!]$ distinguishes $[\![M]\!]$ and $[\![N]\!]$ by adequacy.

Assume M and N are observationally equivalent. Assume that there exists a test $\alpha \in \mathrm{CHO}_{\mathtt{inn},\mathtt{wb}}(A, \mathbf{proc})$ that distinguishes them. By Lemma 6.28, we can choose α so that its reduced form is finite. By applying Lemma 6.37, there exists a term T such that $[\![T]\!]$ is observationally equivalent to α. Because observational equivalence is a congruence, $\alpha \circledcirc [\![M]\!] \simeq_{\mathrm{may}} [\![T\, M]\!]$ and $\alpha \circledcirc [\![N]\!] \simeq_{\mathrm{may}} [\![T\, N]\!]$. Then, applying adequacy, we deduce that T distinguishes M and N which is absurd. □

Extending this result to must equivalence is non-trivial because (a) it is not clear that finite tests suffice and (b) we need somehow to ensure that everyone is equivalent to a strategy verifying properties (1) and (2) of Proposition 6.15.

This full abstraction result tells us that that our notions of innocence and well-bracketing are restricted enough. However we believe that the main argument in favour of these conditions is Theorem 6.20, and its corollary Theorem 6.23 that with innocence and well-bracketing, higher-order behaviours are not exotic.

Part 3

Relaxed impurity

In this part, we are interested in modelling programming languages exhibiting shared memory concurrency. In such languages, there is a way to start several computations in parallel, and these parallel computations can communicate by means of a shared memory. Typically, there is a set of "global variables" that each computation can read and modify. Most semantics for these languages are based on traces. Using the framework of Part 1, we propose to explore the space of partial order models of these behaviours. We believe that such partial-orders models have something to contribute to the semantics of shared-memory concurrency, yet this is space that is barely explored in the literature.

Related work. Shared-memory concurrency has an interesting history from the point of view of semantics. One of the first advocates for a formal account of shared-memory programs was Lamport [**Lam79**] who defined the first mathematical model of execution for such programs, called *sequential consistency* (SC). In this model, the memory is represented as a server that communicates with the threads. Threads submit requests to write and read on global variables, and the role of the memory is to sequentialize these concurrent accesses in a consistent way. Most of the semantic work (denotational in particular) focused on providing models for this paradigm: *eg.* Brookes [**Bro96b**] proves full-abstraction of a trace-based model for Parallel Algol, a prototypical language featuring shared-memory concurrency.

Most reasoning techniques (program logics) and models have assumed this model (and still do, for the most part), even though Lamport already remarks at the time that this model, albeit rather simple to understand and reason with, would an induce important performance penalty on any implementations willing to abide by this specification. As a result, hardware implementing SC is rather a curiosity nowadays and mainstream technology (phones, laptops, ...) features processors breaking away from this model to achieve adequate performance.

This went a long time underspecified, and recently the semantics community has started showing interest in providing models for those architectures that break SC [**NSS**$^+$]. This led to a long series of work to try and understand the existing architectures and formalize the existing specifications provided by hardware manufacturers that were often ambiguous or contradicting observed behaviours.

A few families of architectures were studied: TSO (and its variant) used for instance by Intel, and ARM/POWER, used for instance in most phones nowadays. Two kinds of models emerged. On the one hand, operational (eg. [**OSS09, SSA**$^+$**11**]) models based on the description of the architecture as an abstract machine, and execution is represented as transitions on those states. On the other hand, axiomatic models [**MMS**$^+$**12, AMT14**] axiomatize valid executions via relational algebra.

The two approaches have their qualities: operational semantics models tend to be easier to understand as closer to the machine, whereas axiomatic semantics tend to be more easily experimented with (in particular with the `cat` tool [**AMT14**]) and better for verification purposes as it represents executions by partial-orders.

However, there is very little work on denotational semantics of such hardware. To our knowledge, the only work building on tools of denotational semantics is [**JPR12**] which presents a trace-based model for TSO. More generally, there is very little work on compositionality for weak memory models (operational and axiomatic models are not compositional by design).

In this part, we would like to provide *an* application to the theory we have been developing in the earlier parts of the thesis, to design new models for these specifications. We focus on TSO, which is a simple yet relevant weak memory model, and offers enough interesting behaviours to start testing our framework.

In the next chapters, we will build several models, trying to exploit concurrency as much as possible to avoid needless sequentializations. The style of the following chapters will be more prospective, less thorough, as it tries to understand the specifications from the point of view of our models. We believe we can already make interesting points even though this work is preliminary.

Plan of the part.

Chapter 7. In this chapter, we give a model abiding by the TSO specification, inside event structures of a simple assembly language. Because the language is first-order, there is no need for the game semantics machinery to give an interactive and accurate model of it. The chapter defines several models that try to exploit as much as possible the expressive power of event structures to build more concurrent (hence more compact) models.

Chapter 8. We recast the constructions used on event structures in Chapter 7 inside our game semantics framework. We show that using strategies and their composition it is possible to recast the model construction of Chapter 7 to get another point of view. This formulation in terms of strategies represents better the execution of programs on relaxed architectures and allows for simple tuning by simply changing the strategies implementing the base operations. This permits scaling in a simpler way to weaker architectures allowing more reorderings.

CHAPTER 7

Relaxed memory in a first-order setting

The point of this chapter is to study models of shared memory concurrency within the metalanguage of event structures. To do so, we pick a simple specification of shared memory concurrency (TSO) and we explore the space for denotational models in event structures. The main criterion we have in this chapter is to prevent unneeded sequentializations of concurrent memory accesses, in order to get representations of the possible executions as compact as possible.

Outline of the chapter. Section 1 introduces our toy assembly language and an operational semantics respecting the TSO semantics. Section 2 gives a "naive" interpretation in terms of event structures that is correct with respect to the operational semantics (meaning that the traces of the event structure interpretation coincide with those given by the operational semantics). Section 3 explores the space of interpretations of the memory trying to sequentialize as little as possible the memory accesses in order to avoid the event structure from blowing up.

Contribution of the chapter. The main ideas presented in this chapter were published in **[Cas16]**, although the work on the desequentialization of commits is new.

1. An assembly language and its semantics

1.1. Syntax. We first start by introducing an assembly language, whose programs are parallel composition of threads, and its operational semantics. Consider a countable set $\mathcal{V}$ of **global variables**, that threads share, and a countable set $\mathcal{R}$ of **registers** (or **thread-local variables**) denoting variables local to a thread. The language has simple instructions to access the memory. A *program* is decomposed into *threads* which are lists of such instructions:

$e ::= (k \in \mathbb{N}) \mid (r \in \mathcal{R}) \mid \dots$	(*arithmetic expressions*)
$\iota ::=$	(*instructions*)
$\mid r \leftarrow x \mid x := e$	reads and writes
$\mid$ `mfence`	barrier
$t ::=$	(*threads*)
$\mid \epsilon$	
$\mid \iota ; t$	
$\mid$ `if (0 ==` e`) { ` t ` } { ` t ` }`	conditionals
$p ::= t \parallel \dots \parallel t$	(*programs*)

Our arithmetic expressions are very limited but can be extended without changing the model (as long as they are considered up to simplification).

The instruction $r \leftarrow x$ is a *read* of the variable x (onto the register r) and $x := e$ a *write* of the variable x. The `mfence` instruction, a **barrier**, is specific to relaxed memories and controls the propagation of writes to other threads (see Example 7.2). An occurrence of a register r in a write or a conditional is **free** when not preceded by a read instruction to r (as a result, the expression $r \leftarrow x; t$ can be thought of binding r inside t). A thread is **well-formed** if there are no free occurrences of any register. A program is **well-formed** if all its threads are well-formed.

EXAMPLE 7.1 (Store-buffering). A classic example for studying the properties of a multiprocessor architecture is `sb`, **store-buffering**:

$$\begin{array}{l||l} x := 1 & y := 1 \\ r \leftarrow y & s \leftarrow x \end{array}$$

This notation stands for $(x := 1; r \leftarrow y; \epsilon) \parallel (y := 1; s \leftarrow x; \epsilon)$.

In SC semantics, one would expect that at the end of the execution, we have $r = 1$ or $s = 1$. Indeed, one of the writes needs to go first, and the corresponding read will then read 1. In non-SC semantics, it is often possible to observe $r = s = 0$ if threads are allowed to have **write buffers** (or equivalently, are allowed to reorder independent write/read pairs).

The specification we study in this chapter, *Total-Store-Ordering* (TSO) exhibits the outcome $r = s = 0$. This specification is in particular implemented by Intel processors, by means of write buffers. However, the semantics of TSO can also be understood in terms of instruction reordering [**BMS10**].

To allow the user to flush the buffers (or equivalently, to prevent reorderings), the TSO specification provides a specific instruction, `mfence`.

EXAMPLE 7.2 (Fences). To prevent this behaviour from happening, it is necessary to prevent the optimizations of the processors. This is done by means of an instruction called `mfence`. It forces the processor to empty its write buffer (*ie.* makes the writes available to other threads) – or equivalently, from reordering a write *before* the fence with an independent read *after* the fence. For instance, the following variation on the program of the previous example:

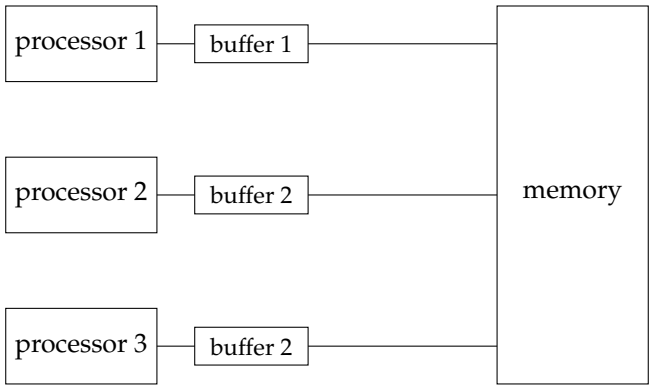
$$\begin{array}{l||l} x := 1 & y := 1 \\ \texttt{mfence} & \texttt{mfence} \\ r \leftarrow y & s \leftarrow x \end{array}$$

does not exhibit the execution leading to $r = s = 0$ (*both* fences are necessary).

Fences are used to prevent the processor from performing optimizations, and as such come with a performance penalty.

1.2. The TSO abstract machine. We describe the standard TSO operational semantics following [**OSS09**], formalized as a transition system over *machine states*.

1.2.1. *Machine states.* A mental model to keep in mind when thinking about TSO machines is the following picture:



Each processor has a (write) buffer that intercepts communication between the processor and the memory. When modelling the system, each component (the processors, the buffers and the memory) needs to be represented. To simplify the modelling, it is easier to group the buffers either with the main memory or with their corresponding processor (as hinted at in the picture). We choose the latter here as it makes modelling easier. The state of the memory is simply an assignment $\mu : \mathcal{V} \to \mathbb{N}$ of global variables. The state of a processor is a pair $(t : \mathfrak{b})$ of a well-formed thread t to execute and a buffer state $\mathfrak{b} \in (\mathcal{V} \times \mathbb{N})^*$ representing the list of writes waiting to be committed to the main memory.

Overall, the machine states have the following shape:

$$\sigma = \langle (t_1 : \mathfrak{b}_1) \parallel \ldots \parallel (t_n : \mathfrak{b}_n) \mathbin{@} \mu \rangle$$

where t_i is a well-formed thread, $\mathfrak{b}_i \in (\mathcal{V} \times \mathbb{N})^*$ a buffer, and $\mu : \mathcal{V} \to \mathbb{N}$ a memory state. If $p = t_1 \parallel \ldots \parallel t_n$ is a program, let $\sigma_p = \langle (t_1 : []) \parallel \ldots \parallel (t_n : []) \mathbin{@} (\_ \mapsto 0) \rangle$ be the initial state of the machine corresponding to p (where $[]$ is the empty buffer), where all variables are set to zero.

1.2.2. *Transitions.* We can now describe the transitions between machine states representing the execution of programs on a TSO architecture. The transitions are given in Figure 1. We comment some of the rules.

(STORE), (IFZ), (IFNZ): By the well-formedness hypothesis, the arithmetic expressions appearing in the first instruction must not contain any register, hence must evaluate to a number. Note that this invariant is preserved in (BUFFERREAD) and (MEMORYREAD) by performing the substitution $t[k/r]$, which goes through t and substitutes free occurrences of r by k.

(COMMIT): This transition can be performed by any thread which has a non-empty buffer, to commit the last entry from said buffer.

(FENCE): A fence can only be performed when the buffer is empty, forcing beforehand the processor to empty its buffer using the (COMMIT) rule.

(MEMORYREAD) and (BUFFERREAD): Those rules are exclusive: when performing a read, either there is an entry in the buffer about the variable we are reading, or there is not. In the former case, the read reads from the most recent entry in the buffer, while in the latter case, it reads from memory.

EXAMPLE 7.3 (Message-passing). Consider the following program `mp`:

$$\begin{array}{l|l} x := 1 & r \leftarrow y \\ y := 1 & s \leftarrow x \end{array}$$

STORE

$$\frac{}{\langle \ldots \parallel (x := k; t_i : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t_i : [(x,k)] +\!\!+ \mathfrak{b}_i) \parallel \ldots @ \mu \rangle}$$

COMMIT

$$\frac{}{\langle \ldots \parallel (t_i : \mathfrak{b}_i +\!\!+ [(x,k)]) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t_i : \mathfrak{b}_i) \parallel \ldots @ \mu[x := k] \rangle}$$

BUFFERREAD

$$\frac{\mathfrak{b}_i = l_1 +\!\!+ [(x,k)] +\!\!+ l_2 \qquad x \text{ does not occur in } l_1}{\langle \ldots \parallel (r \leftarrow x; t_i : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t_i[k/r] : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle}$$

MEMORYREAD

$$\frac{x \text{ does not occur in } \mathfrak{b}_i}{\langle \ldots \parallel (r \leftarrow x; t_i : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t_i[\mu(x)/r] : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle}$$

FENCE

$$\frac{}{\langle \ldots \parallel (\texttt{mfence}; t_i : []) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t_i : []) \parallel \ldots @ \mu \rangle}$$

IFZ

$$\frac{k = 0}{\langle \ldots \parallel (\texttt{if (0 == } k\texttt{) \{ } t \texttt{ \} \{ } u \texttt{ \}} : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (t : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle}$$

IFNZ

$$\frac{k \neq 0}{\langle \ldots \parallel (\texttt{if (0 == } k\texttt{) \{ } t \texttt{ \} \{ } u \texttt{ \}} : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle \longrightarrow \langle \ldots \parallel (u : \mathfrak{b}_i) \parallel \ldots @ \mu \rangle}$$

FIGURE 1. Operational semantics of a TSO machine

One thread writes x (viewed as the data) and then y (viewed as a flag). The other threads first reads the flag, and then the data. The desired outcome is that if the second thread sees the flag to one (ie. $r = 1$), then it sees the update on the data (ie. $s = 1$). As a result, we do not want to observe $r = 1 \wedge s = 0$.

This property is guaranteed by TSO (without having to add barriers), because of a subtle property. In TSO, the buffers are *totally-ordered* meaning that two independent writes are always committed in the syntactic order they appear in the program. This means that x will be always committed first. If the second thread sees the second writes, then it must also see the first one. Moving to partially-ordered buffers where two writes on distinct variables need not be in the same order as in the program yields a weaker architecture, PSO, that allows the outcome $r = 1 \wedge s = 0$.

In general, weaker architectures (ARM/POWER) do not forbid this outcome, as they allow more general reorderings patterns. Programs making use of this programming idiom need to add fences to prevent reorderings.

1.2.3. *Program equivalence.* For $\mu : \mathcal{V} \to \mathbb{N}$, write σ_μ for the final state in memory μ: $\sigma_\mu = \langle \epsilon : [] \parallel \ldots \parallel \epsilon : [] @ \mu \rangle$. Given a program p, we write final(p) for the

set of final memory states reachable from p:

$$\text{final}(p) = \{\mu \mid \sigma_p \longrightarrow^* \sigma_\mu\}.$$

This set characterizes all there is to know about the execution of a closed program. We run the program, and observe the memory in its final state. This leads to a natural contextual equivalence, where the context is given by a program running in parallel. Two programs p, p' are **contextually equivalent** ($p \simeq_{\text{ctx}} p'$) when for all programs q, $\text{final}(p \parallel q) = \text{final}(p' \parallel q)$.

1.3. Labelled semantics. Our previous semantics only gives us the final states reachable by a program but offers no observation of what the program does in the course in the execution. In particular, it cannot be used to reason about the program equivalence defined in the previous section. For these purposes, one often defines a labelled transition system that permits observing the actions of the program via labels on the transitions. Executions become traces of observable events. Since we want a model that is sound for program equivalence (two programs having the same traces are equivalent), the traces need to record enough information about the execution of the program, in *any* context. This means in particular, when issuing a read, we nondeterministically pick a value, since we do not know what the context could have written before.

We could observe every transition made by the machine, however this records *too much* information which is not needed. Informally, processes communicate via memory, so we only need to record those transitions interacting with the memory: the instructions (COMMIT) and (MEMORYREAD), leading to these labels

$$\Sigma ::= \mathtt{c}_{x:=k} \mid \mathtt{R}_{x=k}.$$

A **thread state** is a tuple $(t : \mathfrak{b})$ of a syntactic well-formed thread and a buffer $\mathfrak{b}$. We define a LTS $(t : \mathfrak{b}) \xrightarrow{\ell} (t' : \mathfrak{b}')$ where $\ell \in \Sigma \cup \{\tau\}$ is an action. The rules are given in Figure 2. (The τ label on internal transitions is omitted.) For a thread state $(t : \mathfrak{b})$, write $\text{tr}(t : \mathfrak{b})$ for the set of words $\ell_1 \dots \ell_n \in \Sigma^*$ such that

$$(t : \mathfrak{b}_0) \to^* (t_1 : \mathfrak{b}_1) \xrightarrow{\ell_1} \to^* \dots \xrightarrow{\ell_n} (t_n : \mathfrak{b}_n) \to^* (\epsilon : []).$$

Writing $\star$ for the interleaving of traces, we define the traces of a program $p = t_1 \parallel \dots \parallel t_n$ to be:

$$\text{tr}(p) = \text{tr}(t_1 : []) \star \dots \star \text{tr}(t_n : [])$$

1.4. Consistent traces and outcomes. Among the traces of a program state p, many are not reachable without the help of a context to provide the right values. A trace $t \in \Sigma^*$ is **closed** (or **consistent**) when every read on x of t reads the last value written on x before it in t, or zero if there is none. Closed traces correspond to those traces that can occur if a program is left on its own without any context. Write $\mathcal{C}$ for the set of closed traces.

The **outcome** $\mu(t)$ of a trace $t \in \Sigma^\star$ is the memory state defined as follows:

$$\mu(t)(x) = \begin{cases} k & \text{the last commit } \mathtt{c}_{x:=k} \text{ on } x \text{ in } t \text{ writes } k \\ 0 & \text{otherwise} \end{cases}$$

PROPOSITION 7.4. *Let p be a program. We have:*

$$\mu(tr(p) \cap \mathcal{C}) = \text{final}(p).$$

$$\textsc{Store}\ \frac{}{(x := k; t : \mathfrak{b}) \longrightarrow (t : [(x,k)] +\!\!+ \mathfrak{b})} \qquad \textsc{Commit}\ \frac{}{(t : \mathfrak{b} +\!\!+ [(x,k)]) \xrightarrow{\mathtt{C}_{x:=k}} (t : \mathfrak{b})}$$

$$\textsc{BufferRead}\ \frac{\mathfrak{b} = l_1 +\!\!+ [(x,k)] +\!\!+ l_2 \qquad x \text{ does not occur in } l_1}{(r \leftarrow x; t : b) \longrightarrow (t : b)}$$

$$\textsc{MemoryRead}\ \frac{x \text{ does not occur in } \mathfrak{b}}{(r \leftarrow x; t : \mathfrak{b}) \xrightarrow{\mathtt{R}_{x=k}} (t[k/r] : b)} \qquad \textsc{Fence}\ \frac{}{(\mathtt{mfence}; t : []) \longrightarrow (t : [])}$$

$$\textsc{IfZ}\ \frac{k = 0}{(\mathtt{if\ (0\ ==}\ k\mathtt{)\ \{}\ t\ \mathtt{\}\ \{}\ u\ \mathtt{\}} : \mathfrak{b}) \longrightarrow (t : \mathfrak{b})}$$

$$\textsc{IfNZ}\ \frac{k \neq 0}{(\mathtt{if\ (0\ ==}\ k\mathtt{)\ \{}\ t\ \mathtt{\}\ \{}\ u\ \mathtt{\}} : \mathfrak{b}) \longrightarrow (u : \mathfrak{b})}$$

FIGURE 2. Labelled transition system for thread states

PROOF. Straightforward induction. □

From this, we deduce:

LEMMA 7.5 (Soundness). *For p, p' two programs with $tr(p) = tr(p')$, $p \simeq_{ctx} p'$.*

PROOF. For any context q, we have:

$$\begin{aligned}\text{final}(p \parallel q) &= \mu(\text{tr}(p) \star \text{tr}(q)) \cap \mathcal{C} \\ &= \mu(\text{tr}(p') \star \text{tr}(q)) \cap \mathcal{C} = \text{final}(p' \parallel q). \quad \square\end{aligned}$$

Note that the converse (akin to a full-abstraction result) does not hold, for instance $r \leftarrow x$ is equivalent to the empty program, even though they do not have the same traces. This will be of little importance here, as we simply want to be sure that our labels are rich enough not to lose any behaviours of programs up to this program equivalence.

2. Modelling TSO using event structures

In this section, we investigate how to replay the previous picture in the setting of event structures. What is the counterpart of the labelled semantics? Of consistent traces? In this section, we replace set of traces over the alphabet Σ by Σ-labelled event structure:

DEFINITION 7.6. A Σ-labelled event structure is an event structure E equipped with a labelling function $\lambda : E \to \Sigma$.

A **Σ-labelled partial order** is a Σ-labelled event structure where all finite sets are consistent. As a result, partial orders considered in this chapter satisfy the axioms of event structures; in particular there is a finite number of events below a

fixed event. Usual operations of event structures, in particular parallel composition, extend to labelled event structures.

2.1. Thread semantics. We start by defining the event structure counterpart of the labelled semantics. The labelled semantics of Section 1.3 was defined in terms of a LTS. Even though it is possible to give a definition of the generated set of traces directly by induction on the syntax of threads, this is not completely straightforward. Indeed, if we know the traces for a thread t, to deduce the set of traces of $x := k; t$ is not completely direct. The write instruction should generate a commit event, "later in the future", but how late can it be?

This non-determinism about the actual occurrence of the commits can be elegantly expressed using partial-orders. For instance, in the thread $x := k; r \leftarrow y$, the commit event arising from the write can occur before or after the read. In the labelled semantics, this is represented by having both interleavings. However, in this setting we can represent explicitly the concurrency by declaring that the commit event is concurrent to the read events arising from the read instruction.

To define this new semantics, we need a few operations on event structures.

2.1.1. *Generalized prefix.* To model writes, reads, or fences, given an event structure E (representing the continuation of the program) and a label $\ell \in \Sigma$ (representing the first instruction), we want to form the event structure $\ell \cdot E$ where ℓ occurs first, and then E. However, as we have seen, in the case of write, we do not want the corresponding commit event to occur before *all* the events of E, only a subset. We characterize this subset by their labels, hence we define an operation $\ell \cdot_D E$ where $D \subseteq \Sigma$ is the set of labels of events that need to occur after ℓ.

DEFINITION 7.7. Let $\ell \in \Sigma$ and E be a Σ-labelled event structure. For $D \subseteq \Sigma$, we define $\ell \cdot_D E$ as follows:

Events: $E \cup \{\bot\}$ where $\bot \notin E$.
Causality: The transitive and reflexive closure of

$$\leq_E \cup \{(\bot, e) \mid \lambda_E(e) \in D\}.$$

Consistency: $X \in \mathrm{Con}_{\ell \cdot_D E}$ iff $X \cap E \in \mathrm{Con}_E$.
Labelling: $\lambda(\bot) = \ell$ and $\lambda(e) = \lambda_E(e)$ otherwise.

As a special case, write $\ell \cdot E$ for $\ell \cdot_\Sigma E$.

2.1.2. *Sum of event structures.* When issuing a read, we have seen that we need to nondeterministically pick a value, as there could be a context providing this value. This is done by the **sum of event structures**:

DEFINITION 7.8. For E and E' Σ-labelled event structures, define $E + E'$:

Events, Causality, Labelling: That of $E \parallel E'$
Consistency: $X \in \mathrm{Con}_{E+E'}$ when either $X = \{0\} \times X_0$ with $X_0 \in \mathrm{Con}_E$ or $X = \{1\} \times X_1$ with $X_1 \in \mathrm{Con}_{E'}$

The sum of $E + E'$ is very similar to $E \parallel E'$ except that the two components are not consistent. This operation extends to countable sums, and given $(E_n)_{n \in \mathbb{N}}$ a family of labelled event structures, we write

$$\sum_{n \in \mathbb{N}} E_n := E_0 + E_1 + \dots$$

2.1.3. *Causality of commits.* Before being able to give the inductive definition of our semantics, there is one aspect we need to understand. When calculating the semantics of $x := k; t$, which events of the semantics of t depend on the commit event generated by the write instruction? This is *not* made explicit by our LTS. However, we see that:

(1) Any event after a fence needs to occur after the commit, since the fence forces the commit to be issued before continuing.
(2) Since the buffers are committed in order, any commit arising from a later write needs to be performed later as well
(3) External read (ie. from the memory) on x can also occur only after the write has been committed: as long as the write on x is in the buffer, only internal reads can be performed on x.

Condition (2) and (3) are easily expressed as a set of labels, but condition (1) is not. This forces us to add an event for fences, hence our labels become:

$$\Sigma_i = \Sigma \mid \texttt{mfence}$$

2.1.4. *The inductive definition.* We can now define inductively our semantics, as a Σ_i-labelled event structure. Since it only deals with threads in isolation (without any synchronization), we call it the **thread semantics**, written $\mathscr{T}(t)$. It will be parametrized over a buffer $\mathfrak{b} : \mathcal{V} \rightharpoonup \mathbb{N}$ which is a partial map from variables to values. We will see that we do not need to store the full list.

It is defined as follows:

Empty thread: $\mathscr{T}(\epsilon) = \emptyset$ (the empty event structure)

Fences: Fences issue a `mfence` event and then carry on:

$$\mathscr{T}(\texttt{mfence}; t)(\mathfrak{b}) = \texttt{mfence} \cdot \mathscr{T}(t)(\bot),$$

where $\bot$ is the never defined partial function: indeed, after a fence we know that the buffers are empty.

Reads: The semantics of reads depends on the buffer:

$$\mathscr{T}(r \leftarrow x; t)(\mathfrak{b}) = \begin{cases} \displaystyle\sum_{n \in \mathbb{N}} \mathtt{R}_{x=n} \cdot \mathscr{T}(t[n/r])(\mathfrak{b}) & \mathfrak{b} \text{ undefined at } x \\ \mathscr{T}(t[k/r])(\mathfrak{b}) & \mathfrak{b}(x) = k \\ \quad \displaystyle + \sum_{n \in \mathbb{N}} \mathtt{R}_{x=n} \cdot \mathscr{T}(t[n/r])(\mathfrak{b} \setminus x) & \end{cases}$$

As for the labelled semantics, there are two cases, whether there is an entry in the buffer for x or not. If there is not, we have no choice but to issue a read request to the memory. Otherwise, there are two choices: either we perform an internal read, or, we decide to remove the entries on x in the buffer (committing them) and issue a request.

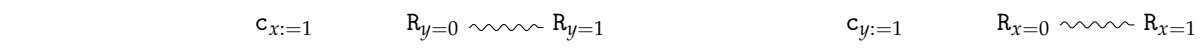
Writes: Let $D_x = \{\texttt{mfence}, \mathtt{c}_{y:=k}, \mathtt{R}_{x=k} \mid y \in \mathcal{V}, k \in \mathbb{N}\}$. We let:

$$\mathscr{T}(x := k; t)(\mathfrak{b}) = \mathtt{c}_{x:=k} \cdot_{D_x} \mathscr{T}(t)(\mathfrak{b}[x := k]).$$

Conditionals: We simply check the condition:

$$\mathscr{T}(\texttt{if (0 == } k\texttt{) \{ } t \texttt{ \} \{ } u \texttt{ \}})(\mathfrak{b}) = \begin{cases} \mathscr{T}(t)(\mathfrak{b}) & (k = 0) \\ \mathscr{T}(u)(\mathfrak{b}) & (k \neq 0) \end{cases}$$

In this definition, we wrote $\mathfrak{b}[x := k]$ for the partial function defined at x with value k and behaving as $\mathfrak{b}$ elsewhere; and $\mathfrak{b} \setminus x$ for the partial function undefined at x and behaving as $\mathfrak{b}$ elsewhere. For instance, the interpretation $\mathscr{T}(\mathtt{sb})([])$ of `sb` (Example 7.1) is:



Notice that the syntactic dependences between the writes and the next read have been broken as they are not preserved by the processors.

2.1.5. *Correctness.* We now show that our semantics is correct: the traces of $\mathscr{T}(t)$ coincide with the traces generated by the LTS. However, $\mathscr{T}(t)$ includes extraneous `mfence` events. We hide them via the projection of event structures:

$$\overline{\mathscr{T}}(t) = (\mathscr{T}(t)([])) \downarrow \lambda^{-1}(\Sigma)$$

where $[]$ is the partial map nowhere defined.

First, we formally define traces of an event structure.

DEFINITION 7.9 (Trace of an event structure). Let $\mathbf{q}$ be a Σ-labelled partial order. A trace of $\mathbf{q}$ is a sequence $\lambda(e_1)\ldots\lambda(e_n)$ such that

$$\emptyset \xrightarrow{e_1} \subset \{e_1\} \ldots \xrightarrow{e_n} \subset \{e_1, \ldots, e_n\} = \mathbf{q}$$

is a covering chain of $\mathbf{q}$. The set of traces of $\mathbf{q}$ is written $\mathrm{tr}(\mathbf{q})$. We write $\mathrm{tr}(E)$ for $\bigcup_{x \in \mathscr{C}(E)} \mathrm{tr}(x)$ for a Σ-labelled event structure E.

PROPOSITION 7.10. *Let t be a well-formed thread. We have:*

$$tr(\overline{\mathscr{T}}(t)) = tr(t : []).$$

PROOF. To show this statement, we need to show a more general statement. Given a buffer state $\mathfrak{b} = [(x_1, k_1), \ldots, (x_n, v_n)] \in (\mathcal{V} \times \mathbb{N})^*$, we write

$$t_{\mathfrak{b}} = (x_1 := k_1; \ldots; x_n := k_n)$$

By induction on t, we show for every buffer state $\mathfrak{b} \in (\mathcal{V} \times \mathbb{N})^*$

$$\mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; t)) = \mathrm{tr}(t : \mathfrak{b}).$$

Conditionals: straightforward

Fences: Since fences force earlier commits to occur, we have

$$\begin{aligned} \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; \mathtt{mfence}; t)) &= \mathtt{C}_{x_1:=k_1} \cdot \ldots \cdot \mathtt{C}_{x_n:=k_n} \cdot \overline{\mathscr{T}}(t) \\ &= \mathrm{tr}(\mathtt{mfence}; t : \mathfrak{b}) \end{aligned}$$

Writes: We have:

$$\begin{aligned} \mathrm{tr}(x := k; t : \mathfrak{b}) &= \mathrm{tr}(t : \mathfrak{b} +\!\!+ [(x, k)]) \\ &= \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b} +\!\!+ [(x,k)]}; t)) \\ &= \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; x := k; t)) \end{aligned}$$

Reads: We prove this result by a sub-induction on $\mathfrak{b}$. If $\mathfrak{b}$ is empty, the result is trivial. Otherwise, write $\mathfrak{b} = \mathfrak{b}' +\!\!+ [(y, k)]$.

If x does not occur in $\mathfrak{b}$, we have:

$$\begin{aligned}
&\mathrm{tr}(r \leftarrow x; t : \mathfrak{b}) \\
&= \mathtt{c}_{y:=k} \cdot \mathrm{tr}(r \leftarrow x; t : \mathfrak{b}') \cup \bigcup_{n \in \mathbb{N}} \mathtt{R}_{x=n} \cdot \mathrm{tr}(t[n/r] : \mathfrak{b}) \\
&\qquad \{ \quad \text{Induction on } \mathfrak{b} \text{ and } t \text{ respectively} \quad \} \\
&= \mathtt{c}_{y:=k} \cdot \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}'}; r \leftarrow x; t)) \cup \bigcup_{n \in \mathbb{N}} \mathtt{R}_{x=n} \cdot \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; t[n/r])) \\
&\qquad \{ \quad \text{Simple examination} \quad \} \\
&= \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; r \leftarrow x; t))
\end{aligned}$$

The last equality follows from x not occurring in $\mathfrak{b}$, $y \neq x$, and the event $\mathtt{c}_{y:=k}$ being concurrent to the read events on x.

If x occurs in $\mathfrak{b}$ with value k, we have:

$$\begin{aligned}
\mathrm{tr}(r \leftarrow x; t : \mathfrak{b}) &= \mathtt{c}_{y:=k} \cdot \mathrm{tr}(r \leftarrow x; t : \mathfrak{b}') \cup \mathrm{tr}(t[k/r] : \mathfrak{b}) \\
&\qquad \{ \quad \text{Induction on } \mathfrak{b} \text{ and } t \text{ respectively} \quad \} \\
&= \mathtt{c}_{y:=k} \cdot \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}'}; r \leftarrow x; t)) \cup \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; t[k/r])) \\
&= \mathrm{tr}(\overline{\mathscr{T}}(t_{\mathfrak{b}}; r \leftarrow x; t))
\end{aligned}$$

Once again, the last line is a careful examination of the shape of

$$\overline{\mathscr{T}}(t_{\mathfrak{b}}; r \leftarrow x; t). \quad \square$$

2.2. Causal justification of the memory. In the world of traces, to recover the correct executions from the labelled semantics, it is enough to consider the intersection of the traces from the program with a set of traces that we consider exhibiting a consistent memory behaviour. In the end, we get a set of traces describing the valid executions of a program.

In this partial-order world, we can do better and provide a causal justification for the behaviour of the memory. For instance, in the store buffering example, we know that $\mathtt{R}_{x=1}$ can occur, but only after $\mathtt{c}_{x:=1}$ has occurred (since it is an external load). More precisely, we would like to sum up the memory behaviour of a closed program with an event structure.

In our setting, we can look at memory behaviours that do not completely sequentialize memory accesses. Intuitively, the purpose of the memory is to ensure that the threads all see a consistent sequentialization of the memory accesses. However, as long as the memory does not get caught, ie. threads observing an inconsistency, all accesses do not need to be actually sequentialized.

2.2.1. *Sets of execution versus event structures.* In traces, it is easy to describe the consistent memory behaviours, leading to the notion of *consistent traces*. Such a notion can be defined *inductively* (at each step, we allow any write, or a *consistent* read, ie. a read that reads the last written value on the same variable before in the trace) or *globally* (a trace is consistent when it does not contain any inconsistent read). To get the consistent executions of a closed programs, we then intersect the labelled semantics with consistent traces.

Event structures are easily built inductively (as for the thread semantics), however the behaviour of a concurrent memory is not easily defined by induction: it is more easily described as a set of executions (partial orders) satisfying a global

correctness criterion. Remember that any event structure induces a set of partial orders $\mathscr{C}(E)$. If a set of partial orders is closed under prefix (intuitively if correctness is "local enough") then we can go back to an event structure by means of prime constructions (Definitions 7.16 and 7.23).

As a result, in the rest of chapters, we will formalize memories as sets of partial orders *first*, and then unfold these sets to event structures. Along the way, we will need to understand the structure of the sets of partial orders we consider, which leads to the notion of *rigid families* (Definition 7.15), the key notion that makes the development go through.

The rest of the chapter is dedicated to building more and more concurrent memories (Definitions 7.11, 7.29 and finally 7.33) which are all equivalent (*i.e.* given a program p, they all predict the same final states), but differ by the amount of sequentializations they impose on concurrent memory accesses. In all cases, we build first a set of partial-order, and then go back to event structures as defining directly the event structures is a difficult task.

To illustrate this methodology of going back and forth between sets of executions and event structures and introduce the theoretical tools needed, we start with a simple "memory", which is still very much sequential, before trying to see how to improve it (Section 3).

2.2.2. *Consistent partial-orders.* Consistent traces generalize naturally to consistent (Σ-labelled) partial-orders:

DEFINITION 7.11 (Consistent partial-order). A Σ-labelled partial order $\mathbf{q}$ is **consistent** when it satisfies:

(1) Two events (reads or commits) on the same variable in $\mathbf{q}$ are comparable for $\leq_{\mathbf{q}}$,
(2) If $e \in \mathbf{q}$ has label $\mathtt{R}_{x=k}$, then the latest write on x below e (well-defined because of (1)) writes k to x.

This definition is quite conservative; as it asks that the history at each variable is linear. We will see later on that we can do better. In the meanwhile, we have:

LEMMA 7.12. *Traces of a consistent Σ-labelled partial order are consistent.*

PROOF. Straightforward (and corollary of Lemma 7.30 proved later). □

Write $\mathscr{Q}_c$ for the collection of consistent partial-orders. Given a program p, to combine $\overline{\mathscr{T}}(p)$ with $\mathscr{Q}_c$, we look at causal enrichments of $\mathscr{T}(p)$ that are in $\mathscr{Q}_c$:

DEFINITION 7.13 (Causal enrichment). A Σ-labelled partial order $\mathbf{q}$ is a **causal enrichment** of another partial order $\mathbf{q}'$ (notation $\mathbf{q} \trianglelefteq \mathbf{q}'$) when they have the same support, and the identity map $\mathbf{q} \to \mathbf{q}'$ is a monotonic map.

We write $\triangleleft$ for the strict version of $\trianglelefteq$: $\mathbf{q} \triangleleft \mathbf{q}'$ when $\mathbf{q} \trianglelefteq \mathbf{q}'$ and $\mathbf{q} \neq \mathbf{q}'$.

Remark that when $\mathbf{q} \trianglelefteq \mathbf{q}'$, we have $\mathrm{tr}(\mathbf{q}') \subseteq \mathrm{tr}(\mathbf{q})$.

Given $\mathbf{q}$ a Σ-labelled partial order, a **causal justification** for $\mathbf{q}$ is a partial order $\mathbf{q}' \in \mathscr{Q}_c$ such that $\mathbf{q} \trianglelefteq \mathbf{q}'$. It is minimally so when there exists no $\mathbf{q} \triangleleft \mathbf{q}'' \triangleleft \mathbf{q}'$ with $\mathbf{q}'' \in \mathscr{Q}$. Write $\mathrm{j}(\mathscr{C}(E))$ for the set of $\mathbf{q} \in \mathscr{Q}_c$ such that there exists $x \in \mathscr{C}(E)$ of which $\mathbf{q}$ is a minimal causal justification. Note that x must be unique, since its support is that of $\mathbf{q}$.

2.2.3. *Rigid families.* This set of partial orders has a very specific property: it is closed under prefix.

DEFINITION 7.14 (Prefix ordering). Let $\mathbf{q}, \mathbf{q}'$ be two Σ-labelled partial orders. If the support of $\mathbf{q}$ is included in that of $\mathbf{q}'$, and the identity map $\mathbf{q} \to \mathbf{q}'$ preserves and reflect labels and causality, then $\mathbf{q}$ is a **prefix** (or **rigid embedding**) of $\mathbf{q}'$ (written $\mathbf{q} \hookrightarrow \mathbf{q}'$).

Note that to give a prefix of a partial order $\mathbf{q}$, it is enough (and sufficient) to give a down-closed subset of its support.

DEFINITION 7.15 (Rigid family, [**Hay14**]). A **rigid family** is a set $\mathscr{Q}$ of Σ-labelled partial orders which is downward closed for the prefix ordering (if $\mathbf{q} \hookrightarrow \mathbf{q}' \in \mathscr{Q}$, then $\mathbf{q} \in \mathscr{Q}$).

Any rigid family can be turned into an event structure by means of the **prime construction**:

DEFINITION 7.16 (Prime construction). Let $\mathscr{Q}$ be a rigid family. We construct the event structure $\mathrm{Pr}(\mathscr{Q})$ as follows:

Events: those partial orders in $\mathscr{Q}$ with a top element (called **primes** of $\mathscr{Q}$),
Causality: prefix ordering,
Consistency: a finite subset X of primes of $\mathscr{Q}$ is consistent when there exists $\mathbf{q}' \in \mathscr{Q}$ such that for all $\mathbf{q} \in X$, $\mathbf{q}$ is a prefix of $\mathbf{q}'$.

The fundamental property of this construction is the following:

LEMMA 7.17. *For any rigid family $\mathscr{Q}$, there is an order isomorphism:*

$$\mathscr{C}(Pr(\mathscr{Q})) \cong \mathscr{Q},$$

where the right hand side is ordered by prefix.

PROOF. The isomorphism maps a $\mathbf{q} \in \mathscr{Q}$ to the set $\{\mathbf{q}' \in \mathrm{Pr}(\mathscr{Q}) \mid \mathbf{q}' \hookrightarrow \mathbf{q}\}$. Conversely $x \in \mathscr{C}(\mathrm{Pr}(\mathscr{Q}))$, there exists $\mathbf{q}' \in \mathscr{Q}$ such that elements of x are prefixes of $\mathbf{q}'$. Write $\mathbf{q}$ for the prefix of $\mathbf{q}'$ defined by the subset of $\mathbf{q}'$ corresponding to $\bigcup x$. It is in $\mathscr{Q}$ because $\mathscr{Q}$ is a rigid family. The map $x \mapsto \mathbf{q}$ gives the desired inverse. □

By analogy with event structures, given a collection of partial orders $\mathscr{Q}$, define

$$\mathrm{tr}(\mathscr{Q}) = \bigcup_{\mathbf{q} \in \mathscr{Q}} \mathrm{tr}(\mathbf{q}).$$

2.2.4. *Closed semantics of programs.* We have now every needed to define the semantics of closed programs. Given a program p, we first consider $\overline{\mathscr{T}}(p)$ which contains all the executions p that can occur in an open context: reads of p read from the environment. Since we consider that p is closed, we are only interested in the executions where the reads of p are actually justified by an early write of p: this is captured by the set of executions $\mathrm{j}(\overline{\mathscr{T}}(p))$. From there, we can recover an event structure via the prime construction. Henceforth, we define the closed semantics of p as follows:

$$[\![p]\!] = \mathrm{Pr}(\mathrm{j}(\overline{\mathscr{T}}(p)))$$

This is correct with respect to the trace semantics:

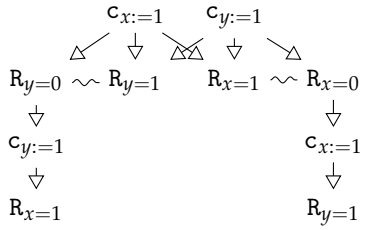
LEMMA 7.18. *For p a program, we have:*

$$tr(p) \cap \mathcal{C} = tr([\![p]\!])$$

PROOF. A trace of $[\![p]\!]$ is both a trace of a configuration x of $\overline{\mathscr{T}}(p)$, and of a consistent causal enrichment of x. By Proposition 7.10 and Lemma 7.12, it belongs to $\mathrm{tr}(p) \cap \mathcal{C}$, as desired.

Conversely if $w \in \mathrm{tr}(p) \cap \mathcal{C}$, there exists $x \in \mathscr{C}(\overline{\mathscr{T}}(p))$ such that $w \in \mathrm{tr}(x)$. Moreover since $w \in \mathcal{C}$, we can regard it as a (linearly ordered) consistent partial order, which is a causal enrichment of x by definition. The justification $x \trianglelefteq w$ might not be minimal but we can extract a minimal one $x \trianglelefteq \mathbf{q} \trianglelefteq w$. As a result, $\mathbf{q}$ corresponds to a configuration of $[\![p]\!]$, of which w is a trace. □

EXAMPLE 7.19. Computing $[\![\mathtt{sb}]\!]$, the semantics of store-buffering, gives:



The memory sequentializes reads and writes on each variable.

2.3. Interaction with the memory. The operation defined in the previous section captures the intended notion of causal justification. It is however not a very familiar operation; in particular it does not fit very well in the semantic framework presented so far in the thesis. Moreover, it does not quite correspond to what is done in traces: to get the traces of p, we need to *intersect* the labelled semantics of p against a set of valid traces. We would like to define $[\![p]\!]$ in a similar way, arising as the interaction of $\mathscr{T}(p)$ and an event structure describing the memory.

2.3.1. *Product of labelled event structures.* In the first part of the thesis, we have seen how to define the interaction of strategies by means of a pullback-like construction on maps of event structures. This pullback-like construction can be seen as a *product* of maps. To transfer the intuitions from the higher-order setting to this first-order setting, we need to compute the product of labelled event structures. It turns out that the two constructions are extremely similar.

Consider $\mathscr{Q}$ and $\mathscr{Q}'$ two sets of Σ-labelled partial orders. An **interaction state** between $\mathscr{Q}$ and $\mathscr{Q}'$ is a secured bijection $\varphi : \mathbf{q} \simeq \mathbf{q}'$ between partial orders $\mathbf{q} \in \mathscr{Q}$ and $\mathbf{q}' \in \mathscr{Q}'$ such that φ preserves labelling. Remember that in this case, φ inherits a partial order defined as $\prec^*$ where $(s,t) \prec (s',t')$ when $s < s'$ or $t < t'$.

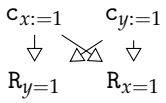
In the previous chapters, to define the interaction of strategies, we did not need to carry explicitly the bijection since it was induced by local injectivity, but here it is part of the structure. Note that, up to isomorphism, the partial order φ is a causal enrichment of $\mathbf{q}$ and $\mathbf{q}'$. Define $\mathscr{Q} \star \mathscr{Q}'$ to be the set of interaction states of $\mathscr{Q}$ and $\mathscr{Q}'$. Since secured bijections are equipped with a partial order relation, $\mathscr{Q} \star \mathscr{Q}'$ is a set of partial orders.

LEMMA 7.20. *If $\mathscr{Q}$ and $\mathscr{Q}'$ are rigid families, then $\mathscr{Q} \star \mathscr{Q}'$ is a rigid family.*

PROOF. Let $\varphi : \mathbf{q} \simeq \mathbf{q}' \in \mathscr{Q} \star \mathscr{Q}'$. Let $\psi \hookrightarrow \varphi$ be a prefix of φ. Since φ is the graph of a bijection, ψ is a graph of a bijection $\mathbf{q}_0 \simeq \mathbf{q}'_0$. Since ψ is downward closed in φ and φ is a causal enrichment (up to isomorphism) of $\mathbf{q}$ and $\mathbf{q}'$, it follows

that $\mathbf{q}_0$ and $\mathbf{q}'_0$ are downclosed in $\mathbf{q}$ and $\mathbf{q}'$ respectively. Since $\mathscr{Q}$ and $\mathscr{Q}'$ are rigid families, this implies that $\mathbf{q}_0 \in \mathscr{Q}$ and $\mathbf{q}'_0 \in \mathscr{Q}'$, hence $\psi \in \mathscr{Q} \star \mathscr{Q}'$. □

Elements of $\mathscr{Q} \star \mathscr{Q}'$ can be seen as compatible superpositions of elements of $\mathscr{Q}$ and $\mathscr{Q}'$. For instance, the configuration of $[\![\mathtt{sb}]\!]$ (Example 7.19)



can be seen as the superposition:

of causal links coming from the program (left) and the memory (right).

As a result, traces of the superposition are traces of both partial orders:

LEMMA 7.21. *Let $\mathscr{Q}, \mathscr{Q}'$ be collections of partial orders (not necessarily closed under prefix). We have:*

$$tr(\mathscr{Q} \star \mathscr{Q}') = tr(\mathscr{Q}) \cap tr(\mathscr{Q}')$$

PROOF. For $\varphi : \mathbf{q} \simeq \mathbf{q}' \in \mathscr{Q} \star \mathscr{Q}'$, we have $\mathrm{tr}(\varphi) \subseteq \mathrm{tr}(\mathbf{q}) \cap \mathrm{tr}(\mathbf{q}') \subseteq \mathrm{tr}(\mathscr{Q}) \cap \mathrm{tr}(\mathscr{Q}')$ as φ causally enriches $\mathbf{q}$ and $\mathbf{q}'$. This proves $\mathrm{tr}(\mathscr{Q} \star \mathscr{Q}') \subseteq \mathrm{tr}(\mathscr{Q}) \cap \mathrm{tr}(\mathscr{Q}')$.

Conversely, let $t \in \mathrm{tr}(\mathbf{q}) \cap \mathrm{tr}(\mathbf{q}')$ with $\mathbf{q} \in \mathscr{Q}$ and $\mathbf{q}' \in \mathscr{Q}'$. Write the covering chains of $\mathbf{q}$ and $\mathbf{q}'$ corresponding to t:

$$\emptyset \xrightarrow{e_1} \cdots \xrightarrow{e_n} \mathbf{q}$$
$$\emptyset \xrightarrow{e'_1} \cdots \xrightarrow{e'_n} \mathbf{q}'$$

By construction, the function $\varphi : \mathbf{q} \to \mathbf{q}'$ mapping e_i to e'_i is a label-preserving bijection. It is easy to see that φ is secured and that $t \in \mathrm{tr}(\varphi) \subseteq \mathrm{tr}(\mathscr{Q} \star \mathscr{Q}')$. □

Given an event structure E, we can always recover a rigid family by considering its domain of configurations $\mathscr{C}(E)$, each configuration being equipped with the order induced by E. This prompts the following definition:

DEFINITION 7.22 (Product of event structures). For E, E' Σ-labelled event structures, their **product** is defined as:

$$E \star E' = \mathrm{Pr}(\mathscr{C}(E) \star \mathscr{C}(E')).$$

Note in passing that this operation is the categorical product in the category of Σ-labelled event structures and label-preserving maps.

From Lemma 7.21, we have $\mathrm{tr}(E \star E') = \mathrm{tr}(E) \cap \mathrm{tr}(E')$ so this operation nicely generalizes the intersection on sets of traces.

2.3.2. *Describing the memory as an event structure.* To complete the picture, we simply have to describe the memory as a single event structure that will interact with $\overline{\mathscr{T}}(p)$. First, notice that consistent partial orders are closed under prefix, hence form a rigid family $\mathscr{Q}_c$. However, there is an awful lot of them since each partial order has an infinite number of isomorphic copies. A **Σ-pomset** is an equivalence class of Σ-labelled partial orders modulo isomorphism. The prefix order on partial orders naturally induces a partial order on pomsets (also called prefix). Moreover, if $\mathscr{Q}$ is a rigid family stable under isomorphism[α] , then one can see it as a set of pomsets, and we will abuse the notation this way. Pomsets can be used to tweak the prime construction:

DEFINITION 7.23 (Prime construction, up to iso). Let $\mathscr{Q}$ be a rigid family closed under isomorphism. The event structure $\mathrm{Pr}_{\cong}(\mathscr{Q})$ is defined as follows:

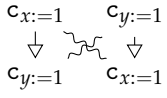
Events: those pomsets $\mathfrak{q} \in \mathscr{Q}$ that have a top element (primes of $\mathscr{Q}$),
Causality: prefix ordering,
Consistency: a finite subset X of primes of $\mathscr{Q}$ is consistent when there exists a pomset $\mathfrak{q}'$ of $\mathscr{Q}$ such that for all $\mathfrak{q}' \in X$, $\mathfrak{q}$ is a prefix of $\mathfrak{q}'$.

As for the usual prime construction, there is map $\mathscr{Q} \to \mathscr{C}(\mathrm{Pr}_{\cong}(\mathscr{Q}))$ analogous to the one of Lemma 7.17 which is surjective, but not injective. However, two partial orders of $\mathscr{Q}$ give rise to the same configuration of $\mathscr{C}(\mathrm{Pr}_{\cong}(\mathscr{Q}))$, if and only if they are isomorphic. As a result, this map induces an isomorphism:

$$\mathscr{Q}/_{\cong} \cong \mathscr{C}(\mathrm{Pr}_{\cong}(\mathscr{Q})).$$

Since $\mathscr{Q}_c$ is stable under isomorphism, it induces an event structure $\mathrm{Pr}_{\cong}(\mathscr{Q}_c)$. However, there are still too many configurations.

EXAMPLE 7.24. Consider the simple program $x := 1 \parallel y := 1$. Because $(\mathsf{c}_{x:=1} \;\; \mathsf{c}_{y:=1})$, $\mathsf{c}_{x:=1} \twoheadrightarrow \mathsf{c}_{y:=1}$ and $\mathsf{c}_{y:=1} \twoheadrightarrow \mathsf{c}_{x:=1}$ are all consistent pomsets, the product $\overline{\mathscr{T}}(p) \star \mathrm{Pr}_{\cong}(\mathscr{Q}_c)$ looks like:



as the unique maximal configuration $\mathsf{c}_{x:=1} \;\; \mathsf{c}_{y:=1}$ of $\overline{\mathscr{T}}(p)$ can synchronize with the three consistent partial orders listed above.

The presence of $\mathsf{c}_{x:=1} \twoheadrightarrow \mathsf{c}_{y:=1}$ and $\mathsf{c}_{y:=1} \twoheadrightarrow \mathsf{c}_{x:=1}$ in $\mathscr{Q}_c$ is superfluous since we know already that $(\mathsf{c}_{x:=1} \;\; \mathsf{c}_{y:=1}) \in \mathscr{Q}_c$. We would like to only keep the elements of $\mathscr{Q}_c$ that have no superfluous causal links, ie. that are **minimal** for $\trianglelefteq$.

Given a rigid family $\mathscr{Q}$, we define

$$\min(\mathscr{Q}) = \{\mathbf{q} \in \mathscr{Q} \mid \forall \mathbf{q}' \in \mathscr{Q}, \mathbf{q}' \trianglelefteq \mathbf{q} \Rightarrow \mathbf{q} = \mathbf{q}'\},$$

the set of $\trianglelefteq$-minimal events of $\mathscr{Q}$. In general it is **not** a rigid family [**CC16**].

[α] There is a slight issue of sizes here, since strictly speaking no rigid families can satisfy this property *and* be a set. In this chapter, we voluntarily remain vague on this aspect so as to avoid technicalities. There are two possible ways of formally handling the problem. The first one is to consider a fixed set of names (eg. $\mathbb{N}$) and ask that the support of all our partial orders should be included in this set. The other one is to define rigid families as presheaves over the category of Σ-labelled partial orders and rigid maps of event structures (ie. maps preserving causality).

LEMMA 7.25. *A consistent $\boldsymbol{q} \in \mathscr{Q}_c$ is minimal if and only if for every causal link $e \twoheadrightarrow e'$ of $\boldsymbol{q}$, e and e' target the same variable.*

PROOF. If $\mathbf{q} \in \mathscr{Q}$ is minimal, and we have $e \twoheadrightarrow e'$ which do not target the same variable, then we can remove this causal link without breaking consistency. Conversely, if the only causal links are between events on the same variable, none can be removed without breaking consistency. □

Hence the structure of minimal consistent partial orders is very simple: it is a parallel composition of linear histories for different variables.

LEMMA 7.26. *The set* $\min(\mathscr{Q}_c)$ *is a rigid family.*

PROOF. Consequence of Lemma 7.25. □

Write $\mathscr{E} = \Pr_{\cong}(\min(\mathscr{Q}_c))$. From Lemma 7.12, it is easy to derive that $\mathrm{tr}(\mathscr{E}) = \mathcal{C}$. As a result,

$$\mathrm{tr}(\overline{\mathscr{T}}(p) \star \mathscr{E}) = \mathrm{tr}(\overline{\mathscr{T}}(p)) \cap \mathcal{C}$$

and $\mu(\mathrm{tr}(\overline{\mathscr{T}}(p) \star \mathscr{E})) = \mathrm{final}(p)$. However, we can show a stronger result:

THEOREM 7.27. *For any program p, there is an (label-preserving) isomorphism:*

$$\overline{\mathscr{T}}(p) \star \mathscr{E} \cong Pr(\mathrm{j}(\overline{\mathscr{T}}(p))) \quad (\textit{which is equal to } [\![p]\!])$$

PROOF. We use Lemma 2.50 to bring the statement at the level of configurations. By Lemma 7.17 applied twice, this amounts to building an iso:

$$\mathscr{C}(\overline{\mathscr{T}}(p)) \star \mathscr{C}(\mathscr{E}) \cong \mathrm{j}(\overline{\mathscr{T}}(p)).$$

Left-to-right. An element of $\mathscr{C}(\overline{\mathscr{T}}(p)) \star \mathscr{E}$ corresponds to a secured bijection $\varphi : x \simeq y$ between $x \in \mathscr{C}(\overline{\mathscr{T}}(p))$ and $y \in \min(\mathscr{Q}_c)$. Remember that φ is a partial-order, and we have a bijection $\varphi \simeq x$. Through this bijection, φ induces a partial order $\mathbf{q}_x$ on the support of x such that $\mathbf{q}_x \cong \varphi$. It is easy to see that $\mathbf{q}_x$ is a causal justification of x, since $x \trianglelefteq \mathbf{q}_x$. The hard part is showing that $\mathbf{q}_x$ is a minimal one.

Assume that we have $x \trianglelefteq \mathbf{q} \trianglelefteq \mathbf{q}_x$ with $\mathbf{q}$ consistent. Assume that there is a causal link $e \twoheadrightarrow_{\mathbf{q}_x} e'$ missing from $\mathbf{q}$. Since $\mathbf{q}$ is consistent, this causal link is between two events targeting different variables. It cannot come from x since $x \trianglelefteq \mathbf{q}$, so it must come from y. This means that y has an immediate causal link between events targeting different variables – this contradicts Lemma 7.25 as y is minimal. This proves that $\mathbf{q}_x \in \mathrm{j}(\overline{\mathscr{T}}(p))$.

As a result, this construction induces a monotonic map

$$\mathscr{C}(\overline{\mathscr{T}}(p)) \star \mathscr{C}(\mathscr{E}) \to \mathrm{j}(\overline{\mathscr{T}}(p))).$$

Right-to-left. By Lemma 7.17, configurations of the right hand side correspond to $\mathbf{q} \in \mathscr{Q}_c$ such that there exists a unique $x \in \mathscr{C}(\overline{\mathscr{T}}(p))$ with $\mathbf{q}$ a minimal causal justification of x. Consider y defined from $\mathbf{q}$ by only keeping the causal links between events targeting the same variable. By construction, $\mathbf{q}' \in \mathscr{Q}$. Moreover, there is a bijection $\varphi : x \simeq y$, which is easily shown secured, such that, as partial orders, we have the isomorphism $\varphi \cong \mathbf{q}$. Mapping $\mathbf{q}$ to φ gives the desired inverse. □

This theorem shows that we can express this causal justification as an interaction against a memory. This point of view will be very fruitful in the next chapter where we move to the world of strategies.

3. Desequentializing memory accesses

In this section, we investigate improvements to the rigid family representing the memory introduced in Section 2.3.2. By forcing the operations on a common variable to be linearly ordered, it leads to a blowup of the final event structure that is not always needed. Indeed, even though every execution will actually exhibit a total order on writes (as guaranteed by TSO), the memory can be smarted and does not to sequentialize all the writes.

EXAMPLE 7.28. Consider the simple program $x := 1 \parallel x := 2$. Its thread semantics is simply $\mathtt{c}_{x:=1} \;\; \mathtt{c}_{x:=2}$, consisting in two concurrent events. After synchronization with the memory, $[\![p]\!]$ is slightly more complicated:

$$\begin{array}{ccc} \mathtt{c}_{x:=1} & \sim & \mathtt{c}_{x:=2} \\ \downarrow & & \downarrow \\ \mathtt{c}_{x:=2} & & \mathtt{c}_{x:=1} \end{array}$$

For this program, moving to event structures did not help, since the resulting event structure does not exhibit any concurrency. We remark however, that this event structure has the same traces as:

$$\mathtt{c}_{x:=1} \qquad \mathtt{c}_{x:=2}$$

which exhibits more concurrency (and as a result is smaller).

The interesting point is that, as long as no one observes in which order the writes have been performed, there is no need to sequentialize them. In this section, we propose to change the notion of consistency in order to relax the assumption that two events operating on the same variable must be comparable.

3.1. Considering only `from-read` and `reads-from`. An easy improvement on $\mathscr{Q}$ is to only order writes and reads on the same variable. Since the end goal is only to exhibit consistent traces, the order of writes does not matter, only their order relative to the reads. This prompts us to replace the first axiom of consistency for partial orders by a weaker version:

DEFINITION 7.29. A Σ-partial order $\mathbf{q}$ is **pre-consistent** when it satisfies:

(1) if $e \in \mathbf{q}$ has label $\mathtt{R}_{x=k}$, then either there are no commits on x in $[e]$ and $k = 0$, or there is a top one, that writes k to x.
(2) a read on x and a write on x are comparable for $\leq_{\mathbf{q}}$

Write $\mathscr{Q}'_c$ for the set of pre-consistent partial-orders.

This is clearly a rigid family. Moreover, it is easy to see that $\mathbf{q} \in \mathscr{Q}'_c$ is minimal when it only contains immediate causal links between a read and a write on the same variable. As a result, $\min(\mathscr{Q}'_c)$ is also a rigid family, and we can consider for a program p, the event structure

$$\overline{\mathscr{T}}(p) \star \mathrm{Pr}_{\cong}(\min(\mathscr{Q}'_c)).$$

We observe that, doing so we have not lost any traces:

LEMMA 7.30. *We have $tr(\mathscr{Q}'_c) = \mathcal{C}$, and as a result*

$$\mu(tr(\overline{\mathscr{T}}(p) \star Pr_{\cong}(\min(\mathscr{Q}'_c)))) = \mathrm{final}(p).$$

PROOF. The final result comes from Lemma 7.21.

If $t \in \mathcal{C}$, then t can be regarded as a Σ-partial order $\bar{t}$. Since it is linearly ordered, it follows that $\bar{t} \in \mathscr{Q}'_c$. Since $t \in \bar{t}$, $t \in \mathrm{tr}(\mathscr{Q}'_c)$.

Conversely assume $t \in \mathrm{tr}(\mathbf{q})$ for $\mathbf{q} \in x$. Assume that t is not consistent. There are two cases.

If there exists a read on x in t not preceded by any commit on x that reads a non-zero value, then there are no commits below this read in $\mathbf{q}$ contradicting axiom (1) of pre-consistency.

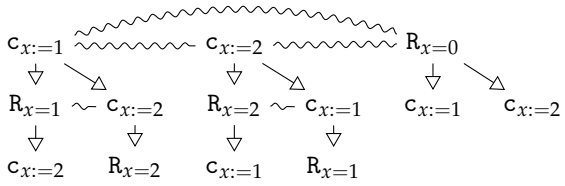
Otherwise there exists a read r on x which reads k but the last commit c in t before r writes k'. Since c appears before r in t, we cannot have $r \leq_{\mathbf{q}} c$. Moreover, as $\mathbf{q}$ is pre-consistent, r and c cannot be concurrent by axiom (2), hence $c \leq r$. Then by axiom (1) of pre-consistency c is below the commit that commits the value seen by r, yet it does not appear in t between c in r: absurd. □

As a result, even though we do not have an isomorphism:

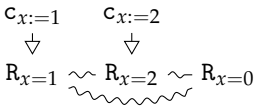
$$\overline{\mathscr{T}}(p) \star \mathrm{Pr}_{\cong}(\min(\mathscr{Q}_c)) \not\cong \overline{\mathscr{T}}(p) \star \mathrm{Pr}_{\cong}(\min(\mathscr{Q}_c)),$$

both event structures have the same traces. At this point, if we want to keep the constraint that we want a rigid family whose traces are consistent, there is not much space for improvement since condition (2) is exactly what ensures the consistency of traces. A little optimization could be to ask that reads and writes on the same variable *with different values* be comparable. But not much else can be done.

3.2. Observing the coherence order. Consider the program $p = x := 1 \parallel x := 2 \parallel r \leftarrow x$. If we compute its semantics according to the previous section, we get:



which features little concurrency. Only when we read $x = 0$, we know that both writes can occur after the read in any order. All these conflicts and sequentialization are necessary to ensure that the read is stuck in between the right writes so as not to observe any inconsistent traces. However, if we are ready to relax these assumptions, we can obtain the following event structure:



This event structure has non-consistent traces, for instance $\mathtt{C}_{x:=1} \cdot \mathtt{C}_{x:=2} \cdot \mathtt{R}_{x=0}$, but it can be reordered to $\mathtt{R}_{x=0} \cdot \mathtt{C}_{x:=1} \cdot \mathtt{C}_{x:=2}$ which is still a trace of the same configuration. As a result, the final outcomes predicted by this event structure (in the end, either $x = 1$ or $x = 2$) agree with the predictions of the operational semantics.

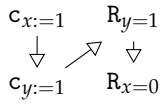
This means that we allow ourselves to build a rigid family whose traces are not all consistent. However, given a trace of a particular partial order in the family, it must be that it can be reordered to a consistent trace without altering the order of commits. This ensures that the final outcomes (memory state) are not affected.

What do we need to fix if we drop axiom (2) of pre-consistency? Let us examine two examples illustrating inconsistent behaviours that become allowed when removing axiom (2).

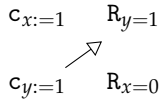
EXAMPLE 7.31. Remember the message-passing example (Example 7.3):

$$\begin{array}{l||l} x := 1 & r \leftarrow y \\ y := 1 & s \leftarrow x \end{array}$$

On TSO, we cannot observe $s = 0 \wedge r = 1$. The partial order $\mathbf{q}_l$:r



does not satisfy axiom (1) of pre-consistency. However, this argument is not enough to ensure that the behaviour does not occur without (2). Nothing forces the intra-thread causality to be considered when checking if a partial order is consistent. As a result, the following is minimal satisfying condition (1) of pre-consistency:



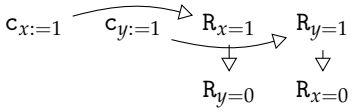
Writing $\mathscr{Q}_{(1)}$ for the rigid family of partial orders satisfying condition (1), the partial order $\mathbf{q}_l$ does arise in $\overline{\mathscr{T}}(\mathtt{mp}) \star \mathrm{Pr}_{\cong}(\min(\mathscr{Q}_{(1)}))$ as the superposition of:

The problem here is due to the memory not being forced to observe the order in which threads send their messages. In reality, the main memory sees in which order each thread sends its requests, so we should force partial orders to observe these intra-thread causal links.

EXAMPLE 7.32. Consider the following program:

$$\begin{array}{l||l||l||l} x := 1 & y := 1 & r \leftarrow x & s \leftarrow y \\ & & r' \leftarrow y & s' \leftarrow x \end{array}$$

Since the commits need to be performed in a fixed order, we cannot observe $r = 1 \wedge r' = 0$ (meaning the commit on x was done first) and $s = 1 \wedge s' = 0$ (meaning the commit on y was done first). However, the following partial order satisfies axiom (1) (and moreover sees the intra-thread causal links):



Because our conditions are so lax, nothing prevents the two writes from being concurrent hence to be observed in different orders by two different threads.

3.3. A new rigid family. To work around those issues, we need to replace condition (2) of pre-consistency by two weaker conditions. One to ensure that consistent partial orders observe enough of the program order, and another to make sure that there are enough causal dependence between writes to prevent inconsistency between observations.

3.3.1. *Adding thread-id information.* To formulate the first condition, we need to add thread-id information to the labels as the conditions rely on them. Given a function $f : \Sigma \to \Sigma'$ and a Σ-labelled event structure $(E, \mathrm{lbl}_E : E \to \Sigma)$, we can form a Σ'-labelled event structure $(E, f \circ \mathrm{lbl}_E)$ that we write $\bar{f}(E)$. We define then an alternative thread semantics which annotates events with the id of the corresponding thread:

$$\overline{\mathscr{T}^{\mathtt{id}}}(t_1 \parallel \ldots \parallel t_n) = \overline{p_1}(\overline{\mathscr{T}}(t_1)) \parallel \ldots \parallel \overline{p_n}(\overline{\mathscr{T}}(t_n))$$

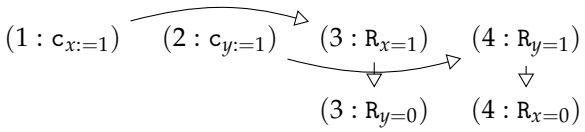
where $p_i(\ell) = (i, \ell)$. Consequently, $\overline{\mathscr{T}^{\mathtt{id}}}(p)$ is a $(\mathbb{N} \times \Sigma)$-labelled event structure. If $k \in \mathbb{N}$ and $\ell \in \Sigma$, we will write $(k : \ell) \in \mathbb{N} \times \Sigma$ for the corresponding label.

3.3.2. *Formulating the conditions.* Having the right labels, we can formulate the conditions for weak consistency. An **independent read/write pair** is a pair of events (e, e') such that one is a read and the other a commit on different variables. Such pairs are not (immediately) ordered by the thread semantics so should not be ordered by the storage semantics either.

DEFINITION 7.33. A $(\mathbb{N} \times \Sigma)$-labelled partial order is weakly consistent when:

(1) if $e \in \mathbf{q}$ has label $\mathtt{R}_{x=k}$, then either there are no commits on x in $[e]$ and $k = 0$, or there is a top one, written $j(e)$, that writes k to x,
(2) two events $e, e' \in \mathbf{q}$ occurring on the same thread are comparable when they are not an independent read/write pair,
(3) if $r \leq_{\mathbf{q}} r'$ with r and r' reads on different variable, and w a commit on the same variable as r', then $j(r) \leq w$.

Axiom (2) prevents the memory from not observing the causality intra-thread, and (3) forces the observation made by the program to be reflected in the causal order. This is enough to kill example 7.32 as the partial order:



is not valid anymore, and cannot be extended to a valid one: $\mathtt{R}_{x=1} \leq \mathtt{R}_{y=0}$ implies that we must have $\mathtt{c}_{x:=1} \leq \mathtt{c}_{y:=1}$, and similarly we should have $\mathtt{c}_{y:=1} \leq \mathtt{c}_{x:=1}$. It is straightforward to check that the collection of weakly consistent partial orders forms a rigid family $\mathscr{Q}_w$.

3.3.3. *Correctness.* We now show correctness of these axioms. We start by establishing an important property of minimal weakly consistent partial orders:

LEMMA 7.34. *Let $\boldsymbol{q} \in \min(\mathscr{Q}_w)$. Assume $e \rightarrowtriangle e'$ where e' is a read on x. Then, one the following is true:*

- *e is a commit on x*
- *e is a read on the same thread of τ.*

PROOF. Assume both propositions are false: e is either a commit on a different variable, or a read on a different thread. Consider the partial order $\mathbf{q}' := (\mathbf{q}, \leq_{\mathbf{q}} \setminus \{(e, e')\}) \trianglelefteq \mathbf{q}$. It is easily seen to be weakly consistent, hence a contradiction. □

LEMMA 7.35. *Let $\boldsymbol{q} \in \min(\mathscr{Q}_w)$ be a minimal weakly consistent partial order, and $t = \ell_1 \cdot \ldots \ell_n$ a trace of $\boldsymbol{q}$. There exists a mapping $f : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that*

(1) $\ell_{f(1)} \cdot \ldots \cdot \ell_{f(n)}$ is a consistent trace of $\boldsymbol{q}$
(2) if ℓ_i and ℓ_j are commits and $i < j$, then $f(i) < f(j)$.

PROOF. Write $\emptyset \xrightarrow{e_1} \cdot \ldots \xrightarrow{e_n} \cdot = \mathbf{q}$ for the covering chain corresponding to t.

We prove the result by induction on the number of inconsistent reads in t (ie. reads that do not read the last committed value according to t). If there are none, t is consistent and f can be the identity.

Otherwise, in t there is a commit $\mathtt{c}_{x:=k}$ (corresponding to e_c in $\mathbf{q}$) and later, a read $\mathtt{R}_{x=k'}$ (corresponding to e_r in $\mathbf{q}$) with $k \neq k'$ with no commits on x in between. Note that we cannot have $e_c \leq e_r$ because this implies $e_c < j(e_r)$ but there are no commits on x in t between e_c and e_r.

If e_r is concurrent with the events $e_c, e_{c+1}, \ldots, e_{r-1}$, we can put e_r before e_c:

$$\emptyset \cdot \ldots \xrightarrow{e_r} \cdot \xrightarrow{e_c} \cdot \ldots \xrightarrow{e_{r-1}} \cdot \xrightarrow{e_{r+1}} \cdot \ldots$$

This covering chain does not permute commits and has one less inconsistent read so we can conclude.

Otherwise, we look at the elements after e_c in t but below e_r in $\mathbf{q}$. There are two cases:

- Either all such events are reads: then all these reads read from earlier commits than e_c in t, so they all can be moved before e_c so as to get a trace of $\mathbf{q}$ with at least one less inconsistent read.
- Or, more interestingly we have in $\mathbf{q}$, $e_{c'}$ a commit on $y \neq x$ immediately followed by a read $e_{r'}$ on y, followed by e_r:

  $$e_{c'}^{\mathtt{c}_{y:=p}} \rightarrowtriangle_{\mathbf{q}} e_{r'}^{\mathtt{R}_{y=p}} \leq_{\mathbf{q}} e_r^{\mathtt{R}_{x=p}}$$

  such that $e_{c'}$ occurs after e_c in t. Because $e_c \not\leq e_r$, e_c and $e_{c'} = j(e_{r'})$ are concurrent. This violates axiom (3) of weak consistency since $e_{r'} \leq e_r$. □

As a result, any outcome coming from a trace of a weakly consistent partial order can be obtained from a consistent trace from the same partial order.

3.3.4. *Semantics of programs.* Can we use this new collection of partial orders to get a more concurrent semantics of our programs? It turns out that it is not so simple. Indeed, the collection $\min(\mathscr{Q}_w)$ is **not** a rigid family: it is not prefix-closed. This is not surprising because axioms (3) and (4) are not "local" in a sense: to know when to order commits, you need to know the future: is the rest of the program going to observe it? For instance, the partial order

$$(1 : \mathtt{c}_{x:=1}) \rightarrowtriangle (2 : \mathtt{c}_{x:=2}) \rightarrowtriangle (3 : \mathtt{R}_{x=2})$$

is in $\min(\mathscr{Q}_w)$ but its prefix $(1 : \mathtt{c}_{x:=1}) \rightarrowtriangle (2 : \mathtt{c}_{x:=2})$ is not minimal in $\mathscr{Q}_w$ as $(1 : \mathtt{c}_{x:=1})\ (2 : \mathtt{c}_{x:=2})$ is also in $\mathscr{Q}_w$. Given any collection of partial orders $\mathscr{Q}$, we can turn it into a rigid family $\downarrow \mathscr{Q}$ by downclosing it for the prefix order:

$$\downarrow \mathscr{Q} = \{\mathbf{q} \mid \exists \mathbf{q}' \in \mathscr{Q}, \mathbf{q} \hookrightarrow \mathbf{q}'\}.$$

However, $\downarrow \min(\mathscr{Q}_w)$ is not adequate since it contains the partial order we want to remove (eg. $(1 : \mathsf{c}_{x:=1}) \nrightarrow (2 : \mathsf{c}_{x:=2})$ as seen above). Fortunately, given a program p the collection of partial orders $\mathscr{C}(\overline{\mathscr{T}^{\mathtt{id}}}(p)) \star \mathscr{Q}_w$ exists, even though it is not a rigid family. Then, we can recover the desired event structure as follows:

$$[\![p]\!]' = \Pr(\downarrow (\mathscr{C}(\overline{\mathscr{T}^{\mathtt{id}}}(p)) \star \mathscr{Q}_w)).$$

This computes the synchronization of the maximal configurations against $\mathscr{Q}_w$, and then down-closes to recover an event structure.

3.3.5. *In practice.* The last couple of definitions might seem a bit complicated. However, they can be thought of (and implemented) more concretely. To compute $[\![p]\!]'$, it suffices to look at the maximal configuration of $\overline{\mathscr{T}^{\mathtt{id}}}(p)$ and for each such configuration, compute the possible causal enrichment that are in $\mathscr{Q}_w$. This is done by picking for each read a commit it reads from, and work out the generated causal constraints to see it this assignment is possible. A single configuration of $\overline{\mathscr{T}^{\mathtt{id}}}(p)$ may lead to several such valid assignments which are the configurations of $[\![p]\!]'$.

In conclusion, we have seen how to use the space offered by event structures to represent memory behaviours that do not sequentialize all accesses on one variable, yet ensure that no inconsistencies are observed. This memory behaviour is *non-local*, it can add causal links between events because of later information about the program. Despite looking odd, this behaviour is actually similar to what happens in weaker memory models such as POWER/ARM where the coherence order is built incrementally as observations are performed. Commits are ordered only when necessary to prevent threads from observing inconsistencies. That is why this analysis, already fruitful in the case of the TSO architecture, is relevant for weaker architectures. However, the models built in this chapter are just examples of the space offered by event structures: it is nowhere close the final word.

In the next chapter, we will see how to recast this in a higher-order setting, by interpreting our language into CHO. We will discuss the advantages of replacing the low-level constructions on event structures and rigid families used here by the composition of strategies.

CHAPTER 8

Relaxed state in a higher-order setting

In this chapter, we continue inside CHO the study started in Chapter 7. More precisely, we want to give a model of the assembly language inside CHO. Because of the cartesian closed structure of CHO, this means that we can also interpret a higher-order extension of this language directly, even though we do not do it explicitly here. We believe however that this possibility is an important advantage of our methodology since it will be crucial to model languages that both exhibit higher order features and relaxed memory features such as Java. However, in this chapter we argue that, even to interpret first-order languages, having a model supporting higher-order is already useful.

The investigations of this chapter are extremely recent, and provide a way to tie together Chapter 7 with the development. The goal of this chapter is twofold:

- recast the interpretation of Chapter 7 inside CHO, transforming the ad-hoc operations on event structures by composition of strategies,
- a way to illustrate the expressive power of CHO by presenting some strategies exhibiting complex concurrent behaviours.

Interestingly, the ideas presented in this chapter came first, and were later simplified to give the framework of Chapter 7. This framework is more elementary (as it does not mention game semantics) but we believe that it is harder to scale to weaker architectures than what is presented here.

Finally, what this chapter presents is more future work than actual research. But we believe that the story presented in this chapter is a nice way to round up the thesis on a less technical and more exploratory note.

Outline of the chapter. Section 1 explains how to see terms of our assembly language as certain terms of PCF augmented with constants to model memory operations. We observe that, because of reorderings, these constants have naturally a higher order type (order 2).

Section 2 interprets these constants inside CHO, and gives another point of view on the interpretation presented in the previous chapter, closer to the execution of programs. The interpretation of programs is then obtained via the composition of strategies which does the heavy lifting of adding the right causal links, according to the strategies interpreting the constants. These constants, along with composition of strategies allow us to abstract away from the very hands on definition of the previous chapter.

Section 3 generalizes the definition of memory behaviours seen in the previous chapter to this setting. Memory behaviours become strategies on a certain arena **mem** representing the protocol between the threads and the memory.

1. PCF$_{\texttt{mem}}$: an extension of PCF with memory operations

The goal of this section is to translate an assembly program (as defined in the previous chapter) to a term of an extension of PCF, PCF$_{\texttt{mem}}$.

1.1. Syntax of the extension PCF$_{\texttt{mem}}$**.** In the previous chapter, we considered programs from a simple assembly language. We explain here how to view such programs as terms of an extension with constants and types of PCF. The resulting extension, PCF$_{\texttt{mem}}$ is given by

$$A, B ::= \ldots \mid \mathbf{proc} \mid \mathbf{mem} \mid \mathbf{mem}_i \qquad (types)$$
$$M, N ::= \ldots \mid \texttt{get}_x \mid \texttt{skip} \mid \texttt{set}_x \mid \texttt{fence} \mid \texttt{hide} \mid \texttt{par} \qquad (terms)$$

where the constants $\texttt{get}_x$ and $\texttt{set}_x$ are indexed over the set of global variables $\mathcal{V}$.

The type **proc** represents the type of effectful computation, just like in IPA **[GM07]** (introduced in Chapter 5), whereas **mem** and $\mathbf{mem}_i$ describe the memory operations available to programs and threads respectively. They do not quite coincide since threads are allowed to issue fences, that are only visible to the thread, and become invisible when considering programs (cf. Section 2.1.5 of Chapter 7).

Intuitively, an element **mem** can be seen as a family $(g_x, s_x)_{x\in\mathcal{V}}$ where $g_x : \mathbb{N}$ is a natural number, that, when evaluated, performs a read on x and returns the outcome, and $s_x : \mathbb{N} \Rightarrow \mathbf{proc}$ is a function that writes the given value in argument to x. Similarly an element of $\mathbf{mem}_i$ can be thought of as a pair $(f, (g_x, s_x)_{x\in\mathcal{V}})$ where $(g_x, s_x)_x$ corresponds to an element **mem** and f : **proc** triggers a fence when evaluated. However, these are simply intuitions (confirmed by their interpretation of arenas given in Section 2.1), and in PCF$_{\texttt{mem}}$, the types **mem** and $\mathbf{mem}_i$ remain abstract and can only be accessed with the corresponding new constants.

The constants $\texttt{get}_x$, $\texttt{set}_x$ and `fence` are used to perform operations on the memory. The constant `skip` represents a closed computation of type **proc** that terminates. The constant `par` is used to start parallel computation, and the constant `hide` is used to convert an element of **mem** to an element of $\mathbf{mem}_i$, intuitively realizing the mapping $(g_x, s_x)_{x\in\mathcal{V}} \in \mathbf{mem} \mapsto (\texttt{skip}, (g_x, s_x)_x) \in \mathbf{mem}_i$. It is used to perform the hiding of fences described in Section 2.1.5 of Chapter 7.

This extension is enough to define a translation from assembly to PCF$_{\texttt{mem}}$. A program p of our assembly language will be translated to a term of PCF$_{\texttt{mem}}$ $\mathtt{m} : \mathbf{mem} \vdash \bar{p} : \mathbf{proc}$, having a free variable $\mathtt{m}$, used to perform the operations on the memory. An individual thread t will be translated to a term $\mathtt{m} : \mathbf{mem}_i \vdash \bar{t} : \mathbf{proc}$.

1.2. Typing the memory operations. Before defining the translation, there is a point that we did not address. What are the types of our new constants?

There is a natural way to convert our assembly language into an extension of PCF$_{\texttt{mem}}$ akin to a state-passing translation. The thread $x := 1; r \leftarrow x; z := r$ would for instance be translated to:

$$m \vdash \texttt{set}_x\, 1\, m\, (\texttt{get}_x\, m\, (\lambda r.\, \texttt{set}_x\, z\, r\, \texttt{skip}))$$

naturally leading to the types

$$\texttt{set}_x : \mathbb{N} \Rightarrow \mathbf{mem} \Rightarrow \mathbf{proc} \Rightarrow \mathbf{proc} \qquad \texttt{get}_x : \mathbf{mem} \Rightarrow (\mathbb{N} \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}.$$

where the last argument represents the continuation (so as to avoid an explicit sequential composition). For the reader familiar with algebraic effects, these explicit

continuations are very much in the spirit of the algebraic presentation of the state monad [**PP02**].

This is not completely satisfactory though. Ultimately our goal is to interpret these constants into CHO in order to deduce an interpretation of PCFmem, and in turn an interpretation inside of assembly code. What can a strategy on $\mathbb{N} \Rightarrow \mathbf{mem} \Rightarrow \mathbf{proc} \Rightarrow \mathbf{proc}$ representing a write on x do? Well it can issue the write, then spawn its continuation. Or spawn its continuation, then issue the write. Or spawn its continuation and in parallel issue the write. None of the choices are enough to implement the subtle reordering behaviours of TSO.

This is due to the type of the continuation being too strict. The only thing $\mathtt{set}_x$ can do about its continuation is start it, and observe when it terminates. However, when interpreting $x := r; t$ in chapter 7, some operations of t should be concurrent to the commit generated by the write instruction, and some should depend on it. The write and t are neither sequentialized, nor parallelized, but "in between". To be able to represent this inspection at the level of strategies, we need to enrich the type of the continuation so that $\mathtt{set}_x$ can observe the memory operations performed by its continuation.

$$\begin{aligned}\mathtt{set}_x &: \mathbf{mem}_i \Rightarrow \mathbb{N} \Rightarrow (\mathbf{mem}_i \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}\\ \mathtt{get}_x &: \mathbf{mem}_i \Rightarrow (\mathbf{mem}_i \Rightarrow \mathbb{N} \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}\\ \mathtt{fence} &: \mathbf{mem}_i \Rightarrow (\mathbf{mem}_i \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}\end{aligned}$$

To interpret the other constants, there are no such subtlety at play. Their types are consequently simpler:

$$\mathtt{par} : \mathbf{proc} \Rightarrow \mathbf{proc} \Rightarrow \mathbf{proc} \qquad \mathtt{hide} : \mathbf{mem} \Rightarrow \mathbf{mem}_i \qquad \mathtt{skip} : \mathbf{proc}$$

1.3. The translation from assembly to PCFmem**.** With these constants in hand, we can define the translation of the assembly code to PCF by induction on the syntax of threads and programs:

Threads: Given a thread t with free registers $r_1, \ldots, r_n$ we define

$$\mathtt{m} : \mathbf{mem}_i, r_1 : \mathbb{N}, \ldots, r_n : \mathbb{N} \vdash \bar{t} : \mathbf{proc}$$

as follows:

$$\begin{aligned}\overline{\epsilon} &= \mathtt{skip}\\ \overline{\mathtt{if\ (0\ ==}\ e\mathtt{)\ \{}\ t\ \mathtt{\}\ \{}\ u\ \mathtt{\}}} &= \mathtt{if}\,(\mathrm{null}\,\bar{e})\,\bar{t}\,\bar{u}\\ \overline{\mathtt{mfence};t} &= \mathtt{fence}\,\mathtt{m}\,(\lambda \mathtt{m}.\,\bar{t})\\ \overline{r \leftarrow x; t} &= \mathtt{get}_x\,\mathtt{m}\,(\lambda \mathtt{m} r.\,\bar{t})\\ \overline{x := e; t} &= \mathtt{set}_x\,\mathtt{m}\,\bar{e}\,(\lambda \mathtt{m}.\,\bar{t})\end{aligned}$$

where $\bar{e}$ is the obvious translation of arithmetic expressions using the corresponding variables for registers, and $\mathrm{null} : \mathbb{N} \Rightarrow \mathbb{B}$ is the test operator of PCF. Interestingly, each operation captures the free variable $\mathtt{m}$ of its continuation. As a result, each operation has access to the dialogue of its continuation with the memory and can tamper with it.

Programs: Programs are translated using `par` to combine the threads, and `hide` to forget about the fences, as the external memory does not need to

see them. If $\mathtt{m} : \mathbf{mem}_i \vdash M : A$, write $\mathtt{hide}\, M$ as a syntactic sugar for $(\lambda \mathtt{m}^{\mathbf{mem}_i}.\, M)\,(\mathtt{hide}\,\mathtt{m})$. Remark that $m : \mathbf{mem} \vdash \mathtt{hide}\, M : \mathbf{proc}$.

$$\overline{t_1 \parallel \ldots \parallel t_n} = \mathtt{par}\,(\lambda \mathtt{m}.\,\mathtt{hide}\,\overline{t_1})\,(\mathtt{par}\ldots(\lambda \mathtt{m}.\,\mathtt{hide}\,\overline{t_n})).$$

For instance, the thread $t = x\mathtt{:=}2; r \leftarrow x; z\mathtt{:=}r$ is translated as

$$\bar{t} = \mathtt{set}_x\,\mathtt{m}\,2\,(\lambda \mathtt{m}.\,\mathtt{get}_x\,\mathtt{m}\,(\lambda \mathtt{m} r.\,\mathtt{set}_z\,r\,(\lambda \mathtt{m}.\,\mathtt{skip}))).$$

2. Thread semantics in a higher-order context

We now recast our thread semantics inside CHO by giving an interpretation of the constants involved in the translation defined in the previous section.

2.1. Arenas for the memory. Since we can translate our assembly code to PCF$\mathtt{mem}$, and we know how to interpret PCF inside CHO, to give a model assembly programs inside CHO, we simply need to interpret the extensions. We start with the types. The type **proc** is interpreted as usual by the arena $\mathbf{run}^- \multimap \mathbf{done}^+$.

In the previous chapter, we had only one event representing a program request, and its answer by the memory. In this setting, because polarity is now explicit (in our case, Program/Memory), we have to explicitly split labels into a program request and a memory answer, to obtain the arenas for **mem** and $\mathbf{mem}_i$.

These arenas represent this protocol from the point of view of the memory. Since the memory should be seen as a server waiting for the program requests, all the minimal moves will be negative (program actions). They represent requests (to write or read a variable) and enable a positive answer (from the memory): acknowledgements in the case of writes, and values in the case of a read.

2.1.1. *The arenas* ***mem****, and* $\boldsymbol{\mathbf{mem}_i}$*.* This protocol is described by an arena $\mathbf{mem}_x$:

The arena for memory is obtained by putting in parallel copies for each variable:

$$\mathbf{mem} = \|_{x \in \mathcal{V}}\, \mathbf{mem}_x.$$

Finally, $\mathbf{mem}_i$ is obtained from **mem** by allowing the program to issue fences. Writing F for the arena $\mathbf{fence}^- \multimap \mathbf{fenced}^+$, we define $\mathbf{mem}_i$ as $F \parallel \mathbf{mem}$. Keeping in mind that parallel composition of arenas represent a product, these definitions coincide with the intuition given in Section 1 as $\mathbf{mem}_x \cong \mathbf{proc} \parallel \mathbb{N}$, and $\mathbf{mem}_i \cong \mathbf{proc} \parallel \mathbf{mem}$ (they only differ by the name of the events).

2.1.2. *Executions of an arena and labelled partial orders.* We can recover the setting of partial orders by considering certain enrichments of configurations $!\mathbf{mem}^\perp$. An **execution** on an arena A is a partial order $\mathbf{q}$ which is a causal enrichment of a configuration x of A such that

(1) the identity labelling defines a receptive and courteous map of event structures to A,
(2) it does not contain two negative moves justified by the same positive move (the memory should answer exactly once to each request).

Two executions $\mathbf{q}$ and $\mathbf{q}'$ are isomorphic when there exists an order isomorphism $\mathbf{q} \cong \mathbf{q}'$ living in $\widetilde{!\mathbf{mem}^\perp}$, *i.e.* preserving labels in **mem**.

LEMMA 8.1. *There is a one-to-one correspondence between:*

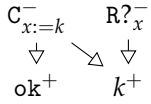
- *Executions on* $!\boldsymbol{mem}^{\perp}$ *up to isomorphism,*
- Σ*-labelled partial orders up to isomorphism.*

PROOF. We consider executions on the dual ($\mathbf{mem}^{\perp}$) because executions are considered from the point of view of the program, describing its actions. Courtesy then ensures that the pairs request/answers are blocks that make sense from a causal point of view: the program can only add causal links from an answer to a request. Receptivity ensures that all blocks are complete (ie. each request has one answer). As a result, we can simply collapse the blocks request/answer in an execution to get a Σ-labelled partial order. □

A similar correspondence can be made between executions on $!\mathbf{mem}_i^{\perp}$ and Σ_i-labelled partial orders.

2.1.3. *Another arena for the memory.* If we wanted to model weaker architectures that perform aggressive reorderings, we might like another representation of this type. In weaker architectures, when the program issues a read request to the main memory, the memory does not answer only with a value but also with the identity of the write this value originates from. This is very important for the processors to know when reordering two reads on the same variable is admissible.

In this case, we would like the answer to the read, to both depend on the read request and the commit request:



but the resulting partial order would not a forest, hence not an arena, making us step out of CHO. To extend CHO to support non-forest like games requires some care, in particular when it comes to single-threadedness, and we leave this extension for future work as this feature will not be needed here.

2.2. Constants for threads. We focus first on the constants used in threads (`skip`, $\texttt{get}_x$, $\texttt{set}_x$, and `fence`). First, the interpretation of `skip` : **proc** is simply the strategy (given by its reduced form) $\mathbf{run}^{-} \multimap \mathbf{done}^{+,0}$ in CHO(**proc**).

The reduced form of the innocent strategy for `fence` is given in Figure 1. In this diagram, we only draw the pointers of positive moves not justified by the initial move, so as to void clutter. This strategy is very simple: it first issues a fence to the external memory (its first argument) and then runs its continuation, forwarding whatever memory operation it does to its external memory. Up to the isomorphism $\mathbf{mem}_i \cong \mathbf{proc} \parallel \mathbf{mem}$, this strategy is the interpretation of

$$\lambda \mathtt{m}.\, \lambda k.\, \pi_1\, \mathtt{m}; k\, \mathtt{m}$$

where $\pi_1\, \mathtt{m}$: **proc** describes the action of performing a fence. The notation r stands for an initial move of **mem**: since `fence` forwards them all to the external memory, there is no need to distinguish them. Similarly, a stands for an answer to r which is automatically propagated back to the continuation.

The strategy for $\texttt{get}_x$ is extremely similar as well, and is depicted in Figure 2. It first performs a read on x, and then spawns its continuation, forwarding directly its memory to the external memory.

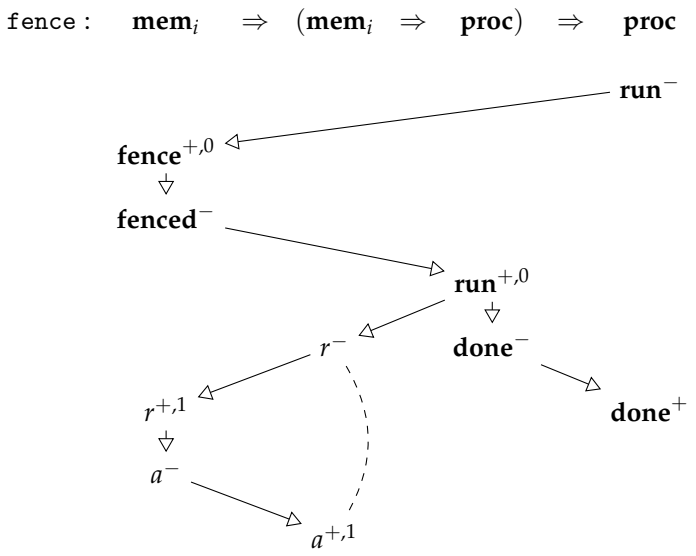


FIGURE 1. A strategy for the constant `fence`

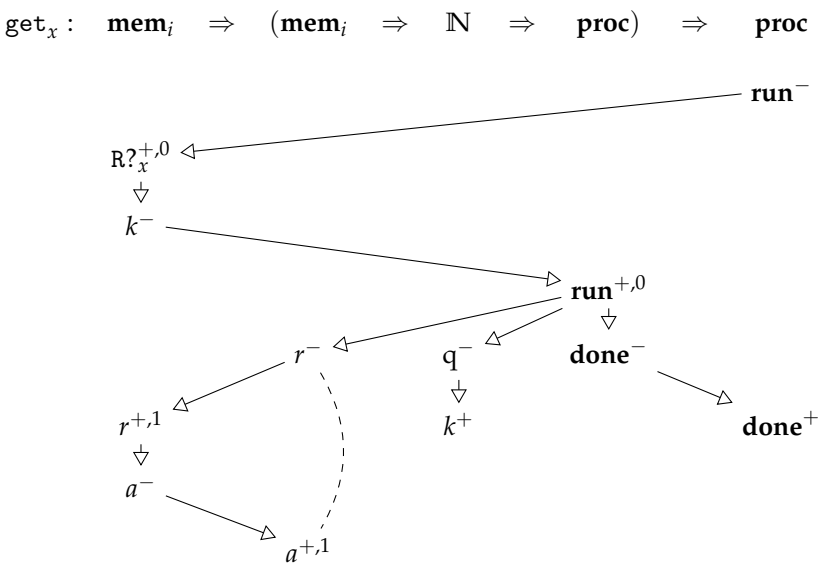


FIGURE 2. A strategy for the constant $\texttt{get}_x$

As for the thread semantics of the previous chapter, things start getting interesting when modelling writes. The previous strategies were sequential, because in TSO the program order from reads and from fences is always respected. As before,

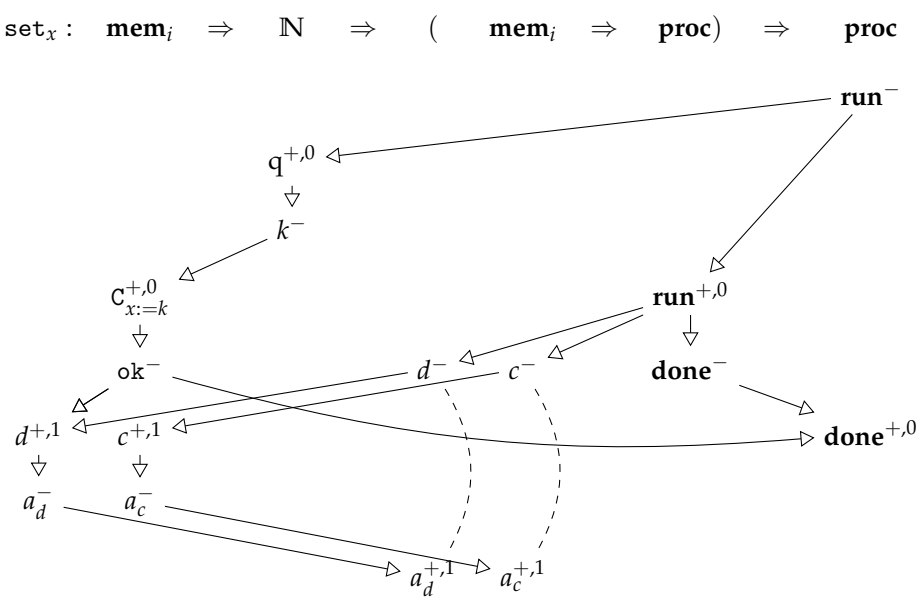


FIGURE 3. A strategy for the constant $\mathtt{set}_x$

we partition the initial moves of $\mathbf{mem}_x$ into two sets:

$$D_x = \{\mathbf{fence}, \mathtt{R?}_x, \mathtt{C}_{y:=k} \mid y \in \mathcal{V}, k \in \mathbb{N}\} \qquad C_x = \{\mathtt{R?}_y \mid y \in \mathcal{V} \setminus \{x\}\}$$

The set D_x contains the requests that should depend on a previous commit on x whereas C_x contains the requests that should be concurrent to previous commits on x. The resulting strategy is depicted in Figure 3.

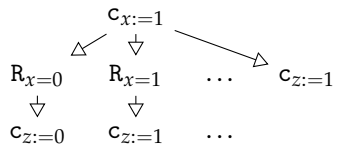
In the diagram, c ranges over elements in C_x and d over elements in D_x. This strategy, in parallel, asks the value of the integer to be written, and issues the corresponding commit request to the memory. In parallel, it runs its continuation. Synchronizations occur when its continuation tries to issue a memory operation that should not be concurrent with the commit in x. In that case, to forward it to the memory, we have to wait until the memory has acknowledged receiving the commit (formalized by the causal link $\mathtt{ok}^- \twoheadrightarrow d^+$). Having put a causal link $\mathtt{ok}^- \twoheadrightarrow \mathbf{run}^+$, we would have a sequential strategy performing no reorderings.

Notice that this strategy is only preinnocent and not innocent, as it is not visible: there is a gcc of a_d^+ that does not go through its justifier, d^-. However, the reduced form can still be expanded non-ambiguously. [α]

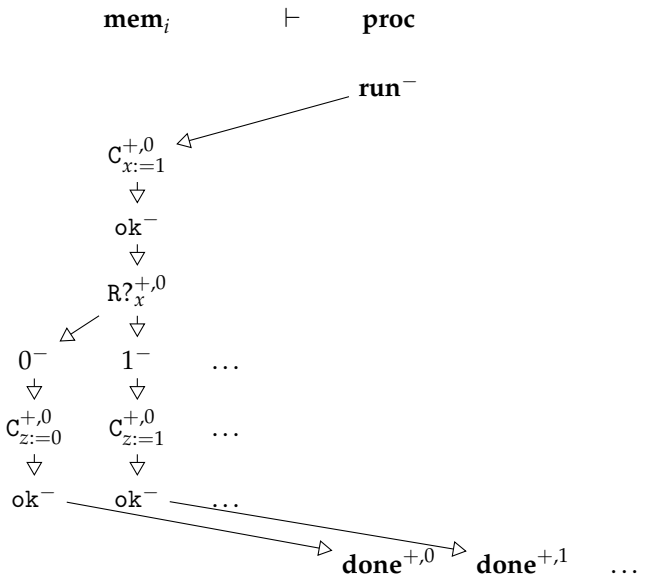
With these three strategies, we can now interpret threads in isolation. Given a thread t we can look at $[\![\bar{t}]\!] \in \mathrm{CHO}([\![\mathbf{mem}_i]\!], [\![\mathbf{proc}]\!])$. Write S_t for the event structure corresponding to $[\![\bar{t}]\!]$. Interestingly, it is conflict-free: as a result S_t is a partial order. We can recover a set of Σ_i-labelled partial orders from S_t by considering all the executions $\mathbf{q}$ which is order-isomorphic through σ to a configuration of S_t, and converting them to Σ_i partial orders.

[α] As, the expansion of reduced form of Chapter 6 actually only needs *pre-innocence* to work

EXAMPLE 8.2. Consider $t = x{:=}1; r \leftarrow x; z{:=}r$. The thread semantics according to the previous chapter gives the following event structure:



We have a commit event on z that is not preceded by a read on x since there is a possibility in this example that the read is satisfied from the buffer entry of the previous write. In that case, no request is sent to the memory. Our strategy $\mathtt{set}_x$ differs as internal loads are not permitted. As a result, the interpretation of $[\![\overline{t}]\!]$ is:



This mismatch is not problematic. As our strategy $\mathtt{set}_x$ implements independent write/read reorderings, this semantics will give the same final states as the one of Chapter 7. The mismatch here is due to the inductive definition being parametrized by a buffer. As a result, the model does not explain how the internal loads are represented at runtime since the buffer information "comes from the sky" (ie. the meta-level). It is possible to represent internal loads in both semantics directly, but the interpretation of writes becomes more complicated as they need to capture (some) later read requests on x and answer the value just written. The corresponding strategy becomes not $\sharp$-innocent as there is a race occurring if the continuation in parallel reads on x and (for instance) writes subsequently on x. Conveniently, this does not occur in the image of our language.

2.3. Constant for programs. To complete the thread semantics, we still need to give the interpretation of two constants: `par` and `hide`, depicted in Figure 4.

The strategy `par` simply spawns its two arguments in parallel and waits for them to terminate before terminating. In the diagram for `hide`, r stands for any

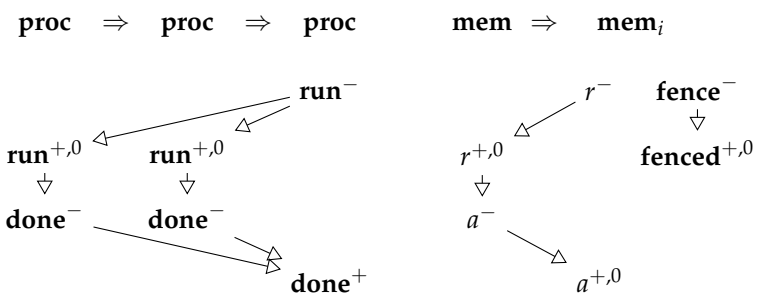


FIGURE 4. Strategies for the constants `par` (left) and `hide` (right)

minimal move of **mem**$_i$ except **fence**. This strategy simply forwards memory operations, and in case of a fence, satisfies it straight away so that the program can continue its execution.

By combining all the strategies, we can then interpret any program p into $\mathrm{CHO}(\mathbf{mem}, \mathbf{proc})$. By extracting the part playing on **mem**, we can recover a set of Σ-labelled partial orders. The resulting set of partial orders does not quite coincide with the configurations of $\overline{\mathscr{T}}(p)$ as hinted above, however both models can be slightly updated to match each other, and both methodologies may be used to represent the same models. As a result, what does the formulation in terms of strategies bring us?

2.4. Strategies as programs. In the previous chapter, our semantics was defined by doing causal surgery on the event structures (especially for interpreting writes). In this setting, we leave the causal surgery to the composition of strategies and focus on the intent of the model: we believe that the formulation of strategies gives a clearer picture of what happens at runtime, than the formulation of the previous chapter. Moreover, strategies are dynamic processes that could be described syntactically. For instance, we can write the complicated strategy for $\mathtt{set}_x$ in the language introduced in Section 1.4.3 of Chapter 6:

$$\begin{aligned}\mathtt{set}_x = \lambda \mathtt{m}.\, \lambda k.\, \lambda c.\ &\text{newsignal } s \text{ in}\\ &(\mathtt{m}(\mathtt{C}_{x:=k}); \text{signal } [s])\\ \parallel\ &c\ (\text{function}\\ &\quad \mid (\mathtt{C}\_:=\_ \mid \mathtt{R}?_x) \text{ as } r \to \text{wait } [s]; \mathtt{m}\ r\\ &\quad \mid \mathtt{R}?_y \to \mathtt{m}\ r)\end{aligned}$$

Here we use a slight extension with ML-like sum types to represent the requests. This rephrases clearly the intent of the write construction: spawn both the commit and the continuation in parallel. Whenever the continuation emits a memory operation that must depend on the commit, wait for the commit before sending the request.

2.5. Towards weaker architectures. Throughout this part we focused on the TSO architecture that features simple reordering patterns. However, when moving to weaker architectures, the setting of the previous chapter becomes slightly harder to update [**Cas16**]. Computing directly the causal structure by induction

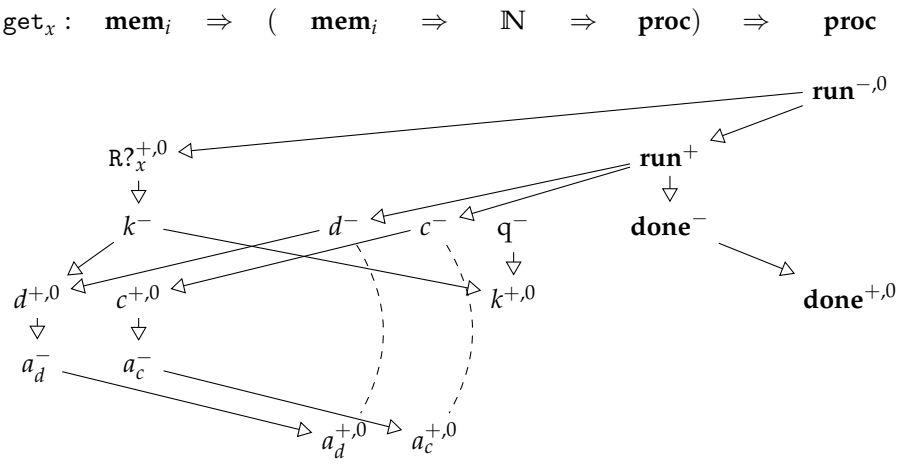


FIGURE 5. An alternative interpretation of $\mathtt{get}_x$ performing more reorderings

is more difficult, because of the problem of *data dependency*. On ARM/POWER, any two instructions targeting distinct variables can be reordered, *except* if there is a data dependency (or *addr* dependency in greater generality): in $r \leftarrow x; y := 1$ we can reorder, but in $r \leftarrow x; y := r$ we cannot, since we need to wait for the value of r to become available. In the world of strategies, it is extremely easy to deal with this phenomenon by changing the strategy for $\mathtt{get}_x$ to that described in Figure 5. This strategy looks very similar to $\mathtt{set}_x$. We assume that the space of memory requests has been partitioned into a set D_x of requests that causally depends on syntactically earlier read on x, and D_x the set of operations that do not. (This depends on the architecture considered). As before c and d range over C_x and D_x respectively. Data dependency is handled by making the answer to q$^-$ (the continuation asking: what is the value of r?) causally depend on the outcome of the read. This mechanism works well and can give an account of the reorderings present in weaker architectures, even with address dependency (in that case, we need to change the arena for the memory but the ideas remain the same). It remains unclear however how to represent speculation and read restarts.

3. Implementing memory

Having seen how to represent the thread semantics inside CHO, we see how to represent the final semantics, with the memory wired in. Since our programs are represented by strategies in CHO(**mem**, **proc**), it seems natural to represent the memory as a strategy $\mathtt{m}$ in CHO(**mem**). However, as we will see, the strategies for memory are never single-threaded: they cannot live in CHO.[β] However, they do live in $\sim$-tCG$^{\cong}_{\circledcirc}$, and we can consider the composition in $\sim$-tCG$^{\cong}_{\circledcirc}$:

$$1 \xrightarrow{\mathtt{m}} !\mathbf{mem} \xrightarrow{[\![\bar{p}]\!]} !\mathbf{proc}$$

[β] This phenomenon already occurs in the model for IPA by Ghica and Murawski [**GM07**].

It gives us a strategy playing on !**proc**, which is not very interesting. For this reason, we consider rather the interaction $[\![\bar{p}]\!] \star \mathtt{m}$ that keeps the internal events in **mem** around. From this interaction, it is easy to extract a set of Σ-labelled partial orders to compare with that from the previous chapter.

3.1. Synchronous memory. From Chapter 7, we have a good idea of the shape of consistent executions. Say that an execution on **mem** is pre-consistent when its collapse to a Σ-labelled partial order is pre-consistent in the sense of Chapter 7. Define the rigid family $\mathscr{Q}_{\mathtt{m}}$ as:

$$\mathscr{Q}_{\mathtt{m}} :=\downarrow \{\mathbf{q} \mid \mathbf{q} \text{ is a pre-consistent execution}\}.$$

The pre-consistent executions are not closed under prefix (hence the prefix closure): in an execution, all requests must be answered, which means that eg. $\mathtt{C}^+_{x:=1}$ is not a execution. The prefix closure allows us to consider "partial executions" where not all requests have been answered yet. Write $\mathscr{E}_{\mathtt{m}}$ for $\mathrm{Pr}(\mathscr{Q}_{\mathtt{m}})$: it inherits a labelling $\mathtt{m} : \mathscr{E}_{\mathtt{m}} \to !\mathbf{mem}$. However, it is not a strategy as it is not courteous. For instance, it contains the following configuration:

$$\mathtt{C}^-_{x:=1} \rightarrowtriangle \mathtt{ok}^+ \rightarrowtriangle \mathtt{R}?^-_x \rightarrowtriangle \mathtt{R}?^+_1$$

which is not courteous because of the link $\mathtt{ok}^+ \rightarrowtriangle \mathtt{R}?^-_x$ is not present in **mem**.

As $\mathtt{m}$ is not a strategy, the composition $[\![\bar{p}]\!] \circledcirc \mathtt{m}$ has no reason to be a strategy (courteous in particular). This is not a problem in this setting, as the interaction $[\![\bar{p}]\!] \star \mathtt{m}$ still exists, and it is unsettling. This is incompatible with interpreting languages where the memory does not come at the end, but it might be necessary to wire in the memory before the whole program is known. An example would be to implement a restriction operator r_x : **mem** $\Rightarrow$ **mem** such that $r_x(\lambda \mathtt{m}.\, \bar{p})$ is a new program where all the actions of x inside p are synchronized together and hidden from the outside world (akin to the `new` construct of IPA). To interpret such a construct, we need r_x to be a proper strategy, and as a result, $\mathtt{m}$ as well.

In this particular case, because $\mathtt{m}$ has an empty domain, $[\![\bar{p}]\!] \circledcirc \mathtt{m}$ is an *essential strategy* in the sense of Chapter 2.

LEMMA 8.3. *Let $\sigma : S \rightharpoonup A$ be a pre-strategy in the sense of Chapter 2 (ie. a partial map of event structure) and $\tau : T \rightharpoonup A^\perp \parallel B$ an essential strategy of $\mathrm{CG}^{\cong}_{\circledcirc}(A, B)$. Then the composition $\tau \circledcirc \sigma$ is in $\mathrm{CG}^{\cong}_{\circledcirc}(B)$.*

PROOF. The proof relies on the characterization of strategies as composing well with copycat (Theorem 2.79). The composition $\tau \circledcirc \sigma$ is an essential strategy if and only if $\mathfrak{a}_B \circledcirc (\tau \circledcirc \sigma)$ is isomorphic to $\tau \circledcirc \sigma$. By associativity, we have:

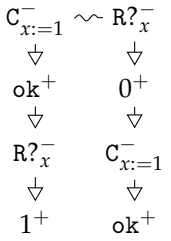
$$\mathfrak{a}_B \circledcirc (\tau \circledcirc \sigma) \cong (\mathfrak{a}_B \circledcirc \tau) \circledcirc \sigma \cong \tau \circledcirc \sigma$$

since τ is an essential strategy. □

Our characterisation of essential strategies allows us here to derive that, at least at the level of event structures, it does not matter that $\mathtt{m}$ is not *quite* a strategy.

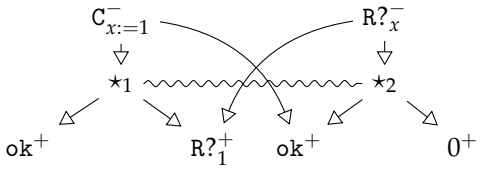
3.2. Asynchronous memory. This does not allow us to build a strategy in $\sim$-$\mathrm{tCG}^{\cong}_{\circledcirc}$ since there is no similar characterization at the level of symmetries.

The problem is that $\mathtt{m}$ is synchronous, enforcing its own order on the event, regardless of what the program does. Consider the following excerpt of the (infinite) event structure $\mathscr{E}_{\mathtt{m}}$:



The minimal conflict at the beginning means that, in a way, this pre-strategy non-deterministically decides what is going to happen.

To recover an essential strategy out of this, we can use again the characterisation of essential strategies, and the fact that copycat is idempotent. As a result, $\bar{\mathfrak{m}} := \mathfrak{a}_{!\mathbf{mem}} \circledcirc \mathfrak{m}$ is an essential strategy, which represents the asynchronous version of $\mathfrak{m}$. For instance, the previous excerpt of $\mathfrak{m}$ becomes in $\bar{\mathfrak{m}}$:



The two neutral events symbolize the races between the two causal histories: if $\star_1$ is played, then the left causal history is chosen, where the commit goes first, and if $\star_2$ is played, the right one is chosen, where the read goes first.

To turn $\bar{\mathfrak{m}}$ into a strategy of $\sim$-tCG$^{\cong}_{\circledcirc}$, we need to build an isomorphism family.

DEFINITION 8.4. A symmetry between two configurations of $\mathscr{E}_{\mathfrak{m}}$ is an order-isomorphism preserving labels in $!\mathbf{mem}$. Write $\widetilde{\mathscr{E}_{\mathfrak{m}}}$ for the set of such symmetries.

LEMMA 8.5. *The pair $(\mathscr{E}_{\mathfrak{m}}, \widetilde{\mathscr{E}_{\mathfrak{m}}})$ is a thin event structure with symmetry that extends $\mathfrak{m}$ into a $\sim$-receptive map of event structure with symmetry, ie. a pre-$\sim$-strategy.*

PROOF. Straightforward verification. □

Since pre-$\sim$-strategies compose, we know that $\bar{\mathfrak{m}} = \mathfrak{a}_{!\mathbf{mem}} \circledcirc \mathfrak{m}$ is a pre-$\sim$-strategy, which is also an essential strategy. So, by definition is $\sim$-strategy, hence $\bar{\mathfrak{m}} \in \sim\text{-tCG}^{\cong}_{\circledcirc}(1, \mathbf{mem})$ as desired.

However, as seen in the previous diagram, $\mathfrak{m}$ and $\bar{\mathfrak{m}}$ are not single-threaded. However, a single-threaded strategy can be recovered if we are ready to change slightly the type of memories.

LEMMA 8.6. *Consider A, B negative arenas. Write*

$$d = [\![\lambda x.\, (\lambda f.\, f\, x)]\!] \in \mathrm{CHO}(A, (A \Rightarrow B) \Rightarrow B).$$

For any strategy negative $\sigma \in \sim\text{-tCG}^{\cong}_{\circledcirc}(!A)$, the composition (in $\sim$-tCG$^{\cong}_{\circledcirc}$) $d \circledcirc \sigma$ is single-threaded.

As a result, $d \circledcirc \bar{\mathfrak{m}}$ is a strategy of $\mathrm{CHO}(\mathbf{mem} \Rightarrow \mathbf{proc}, \mathbf{proc})$, that can be used to interpret a IPA-like construct:

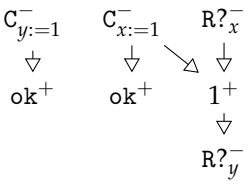
$$\frac{\Gamma, \mathfrak{m} : \mathbf{mem} \vdash M : \mathbf{proc}}{\Gamma \vdash \text{newmemory } \mathfrak{m} \text{ in } M : \mathbf{proc}}$$

that could be used to restrict the access to a memory inside a sub-term.

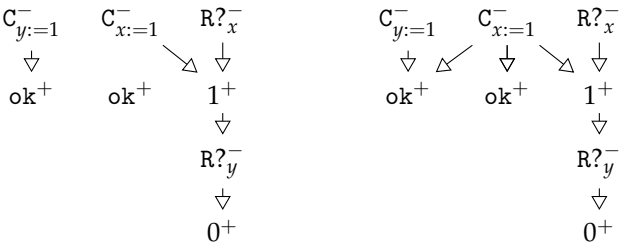
3.3. Towards more concurrent memories. In the last chapter, we have seen that the resulting strategy $\mathtt{m}$ is still more sequential than needed as it sequentializes more that can be needed. It is possible to do better, by trying to only sequentialize writes when the program later actually observe the writes. But this quantification over a possible future makes it impossible to create a rigid family (or in this setting a strategy), since when given two writes, we need to decide *now* whether to order them or not.

A possible way to solve this problem is not to force the strategy implementing the memory to actually put any causal link to order writes, but simply to let them be concurrent. However, the memory keeps as an internal state the partial order justifying that the current execution is correct. Because the partial order does not represent the causality of the execution, but simply a justification for the current execution, it is possible to add causal links in the past.

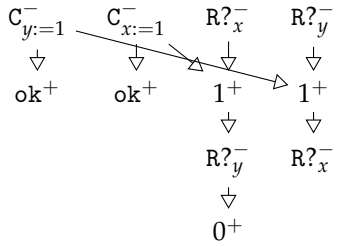
For instance, assume that the current execution is:



This is already valid: there is no need to add causal links for this execution to be weakly consistent. At this point, the memory is allowed to satisfy the read on y in two different ways. It can answer $y = 1$ and make this answer depend on $\mathtt{C}^-_{y:=1}$. It can also answer $y = 0$, but in this case, the execution is not weakly consistent anymore since the two commits are not ordered. Since both commits have already occurred, the memory cell cannot go back in the past and enforce, actually the commit on y depended on the commit on x. So the only thing it can do is to add this causal link in its internal state:



On the left is depicted the actual execution, on the right the internal state of the memory justifying that the execution is correct. The internal state is used by the memory to avoid answering in an inconsistent way. Imagine that, later in the execution, another thread comes along and reads on y and then on x so that the execution is now:



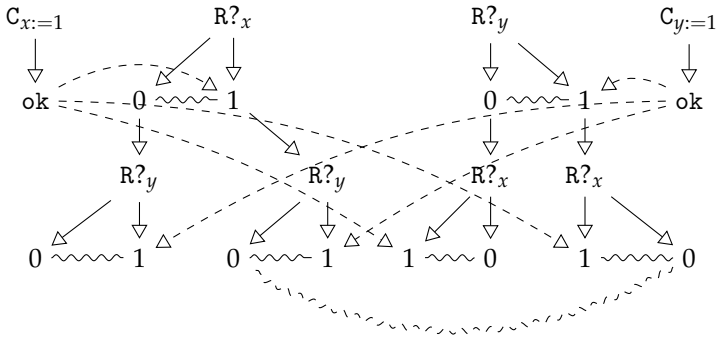
In this case, the memory could be tempted to answer $x = 0$ if its internal state did not remind it that there is a causality from the commit on x to that on y, and reading $x = 0$ would force it to put a causal link in the other direction, resulting in a cycle. So the memory knows that it can only answer $x = 1$ in this context.

It is possible to turn this intuition into an event structure whose events should be pairs of an event of a very relaxed notion of execution (with only intra-thread causal links and links from a commit to the reads loading its value), along with a causal justification that this execution is actually weakly consistent. However the correctness of such a strategy is not completely straightforward to establish.

To give an example of what it can achieve, write $\mathtt{m}'$ for this loosely-defined strategy, and p the following program:

$$x := 1 \;\Big\|\; \begin{array}{l} r_1 \leftarrow x \\ r_2 \leftarrow y \end{array} \;\Big\|\; \begin{array}{l} s_1 \leftarrow y \\ s_2 \leftarrow x \end{array} \;\Big\|\; y := 1$$

Then, then the part of $[\![p]\!] \star \mathtt{m}'$ on $\mathbf{mem}^\perp$ is:



In this diagram, we have represented differently causal links and conflicts coming from $[\![p]\!]$ (solid) and those coming from $\mathtt{m}'$ (dashed). The interesting thing to remark on this diagram is the dashed conflict between "reading 1 and then 0" on one thread, and same outcome on the other thread. Both events cannot be concurrent, since there is no justification for the configuration of $\overline{\mathscr{T}}(p)$ which contains them both, or equivalently, their internal state are incompatible since they disagree on the order between the two commits. This configuration would correspond to inconsistent observations by the threads.

Conclusion and perspectives

To conclude this thesis, we propose perspectives, and leads left open relative to the developments presented throughout the thesis.

First part: concurrency

$\sim$-tCG up to weak bisimulation. In Chapter 2, we construct two related categories, one up to weak bisimulation without any hiding, and one up to isomorphism with some hiding. In the setting with symmetry, we only construct the latter. We believe that having both categories is actually an advantage when modelling programming languages. One can model a language in the category up to weak bisimulation (where there is more space) and deduce a more compact interpretation in the category up to isomorphism. Having both interpretations can be useful to pick the right point of view depending on the result. For instance, the strong link between the operational semantics and the denotational semantics proved in Chapter 4 (Theorem 4.29) could be even strengthened in the setting up to weak bisimulation.

Study a subset of thin concurrent games. Our notion of thin concurrent games is very general. In most applications we are interested in, the symmetry on games is always *local*, ie. the symmetry boils down to an equivalence relation on events. Note that the symmetry on strategies can be non-local as evidenced in Section 1.4.2 of Chapter 6. We believe this case would greatly simplify the setting by making the two sub-symmetries $\mathcal{A}_-$ and $\mathcal{A}_+$ obtainable from $\mathcal{A}$. As a result, in these particular cases, the sub-symmetries may not need to be part of the structure.

Generalize CHO to non-forest arenas. As alluded to in Chapter 8, the fact that objects of CHO must be forest-like games can be problematic when trying to give sophisticated interpretations to the **mem** type (to allow a read answer to be justified by a commit request). We believe that the construction of CHO should follow through if we restrict to games that are not forest but simply single-threaded. In this new setting, **mem** would not be a valid object but $(\mathbf{mem} \Rightarrow \mathbf{proc}) \Rightarrow \mathbf{proc}$ would be, which would be enough to carry out the construction of the model.

Reduced forms and metalanguage for non-innocent strategies. One of the most interesting leads concerning the framework would be to understand the structure of non-innocent strategies. In the innocent case, we know what finite strategies are: those whose reduced form only reacts on a finite number of Opponent answers. The situation for non-innocent strategies is much less clear as their structure is more complicated. This finiteness criterion would allow us to prove definability results for non-innocent fragments by induction, something impossible at the moment.

A related problem is that to find a language corresponding to strategies, supporting a definability result up to isomorphism of strategies (the one presented in Chapter 6 is only up to may equivalence). A preliminary candidate is hinted at in Section 1.4.3 of Chapter 6. As noticed in Chapter 8, this would allow us to easily implement models of programming languages, by simply implementing the strategies interpreting the constructs of the language.

Second part: innocence

A language for concurrent innocence. Related to the issue above, it would be interesting to have a language corresponding to concurrent innocence. It should contain control operators of some sort to allow for non well-bracketed behaviours. This result would help us understand better the situation of concurrent control operators. Is `call/cc` enough, or is there new non well-bracketed behaviours that appear in a concurrent world that are not definable from `call/cc`?

Link of our innocence with the sheaf-theoretic notion. In their line of work, Hirschowitz *et. al.*[**Hir13**] introduced the idea of representing innocence as a sheaf condition on strategies represented as presheaves. This notion was later recast in the λ-calculus by Tsukada and Ong [**TO15**]. We believe that there is a strong link between preinnocence (ie. innocence without locality) and this sheaf condition, since our strategies can be seen as certain presheaves over causal augmentations of the game. It is not clear where locality fits in this picture and technical details need to be carried out.

Generalize the intensional full abstraction result to must equivalence. In Chapter 6, we show intensional full abstraction for may equivalence. We believe the model also supports a similar result for must equivalence, however several obstacles are still in the way. The most important one is that finite test should have the same discriminating power as all tests. For may equivalence, it is easy to show that finite tests suffice, but for must equivalence it is much harder since it is not clear how, given a test discriminating (for must convergence) two strategies, to extract a finite part, still discriminating the strategies.

Third part: relaxed memory

Investigate weaker architectures. We believe our model scales to weaker architectures as the ideas presented in Chapters 7 and 8 explain how to deal with complicated memory cells and sophisticated reorderings. However, the work remains to be done. We have started investigating a thread semantics for POWER in this framework.

Language specifications. We have not mentioned language specifications in this part (C11, Java for instance). They offer new challenges that cannot be dealt with the techniques, in particular closure with respect to compiler optimizations. In operational models, this proved to be a challenge [**KHL$^+$17**] to represent, and we would like to understand how to adapt these techniques to our framework.

Partial synchronizations. In the models presented in Part 3, all the synchronizations occur at the end, when the whole program is known. We believe there is a way to compute these synchronizations incrementally (similarly to what the CCS model in event structures is doing [**Win82**]) as a way to perform synchronization in a compositional way.

Bibliography

[AHM98] Samson Abramsky, Kohei Honda, and Guy McCusker. A fully abstract game semantics for general references. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, June 21-24, 1998*, pages 334–344. IEEE Computer Society, 1998.

[AJM00] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.

[AM99a] Samson Abramsky and Guy McCusker. Full abstraction for Idealized Algol with passive expressions. volume 227, pages 3–42. 1999.

[AM99b] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 431–442, 1999.

[AMT14] Jade Alglave, Luc Maranget, and Michael Tautschnig. Herding cats: Modelling, simulation, testing, and data mining for weak memory. *ACM Trans. Program. Lang. Syst.*, 36(2):7:1–7:74, 2014.

[BC82] G. Berry and Pierre-Louis Curien. Sequential algorithms on concrete data structures. *Theor. Comput. Sci.*, 20:265–321, 1982.

[BMS10] Sebastian Burckhardt, Madanlal Musuvathi, and Vasu Singh. Verifying local transformations on relaxed memory models. In Rajiv Gupta, editor, *Compiler Construction, 19th International Conference, CC 2010*, volume 6011 of *Lecture Notes in Computer Science*, pages 104–123. Springer, 2010.

[Bro96a] Stephen D. Brookes. The essence of parallel algol. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 164–173. IEEE Computer Society, 1996.

[Bro96b] Stephen D. Brookes. Full abstraction for a shared-variable parallel language. *Inf. Comput.*, 127(2):145–163, 1996.

[Cas16] Simon Castellan. Weak memory models using event structures. In David Baelde and Jade Alglave, editors, *Vingt-septième Journées Francophones des Langages Applicatifs (JFLA 2016)*, 2016.

[CC16] Simon Castellan and Pierre Clairambault. Causality vs interleaving in game semantics. In *CONCUR 2016 - Concurrency Theory*, 2016.

[CCRW] Simon Castellan, Pierre Clairambault, Silvain Rideau, and Glynn Winskel. Games and strategies as event structures. *Logical Methods in Computer Science*. Accepted with minor revisions for publication.

[CCW14] Simon Castellan, Pierre Clairambault, and Glynn Winskel. Concurrent Hyland-Ong games. *CoRR*, abs/1409.7542, 2014.

[CCW15] Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of PCF. In *LICS 2015*. IEEE Computer Society, 2015.

[CF05] Pierre-Louis Curien and Claudia Faggian. L-nets, strategies and proof-nets. In C.-H. Luke Ong, editor, *Computer Science Logic, 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3634 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 2005.

[CHLW14] Simon Castellan, Jonathan Hayman, Marc Lasson, and Glynn Winskel. Strategies as concurrent processes. *Electr. Notes Theor. Comput. Sci.*, 308:87–107, 2014.

[Cur92] Pierre-Louis Curien. Observable algorithms on concrete data structures. In *Proceedings of the Seventh Annual Symposium on Logic in Computer Science (LICS '92), Santa Cruz, California, USA, June 22-25, 1992*, pages 432–443. IEEE Computer Society, 1992.

[DBL05] *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings*. IEEE Computer Society, 2005.

[EHS13] Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. Fully-abstract concurrent games for pi. *CoRR*, abs/1310.4306, 2013.

[EHS15] Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. An intensionally fully-abstract sheaf model for pi. In Lawrence S. Moss and Pawel Sobocinski, editors, *6th Conference on Algebra and Coalgebra in Computer Science, CALCO 2015, June 24-26, 2015, Nijmegen, The Netherlands*, volume 35 of *LIPIcs*, pages 86–100. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[FM05] Claudia Faggian and François Maurel. Ludics nets, a game model of concurrent interaction. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings* **[DBL05]**, pages 376–385.

[FP09] Claudia Faggian and Mauro Piccolo. Partial orders, event structures and linear strategies. In *TLCA '09*, volume 5608 of *LNCS*. Springer, 2009.

[GM07] Dan R. Ghica and Andrzej S. Murawski. Angelic semantics of fine-grained concurrency, 2007.

[Hay14] Jonathan Hayman. Interaction and causality in digital signature exchange protocols. In Matteo Maffei and Emilio Tuosto, editors, *Trustworthy Global Computing - 9th International Symposium, TGC 2014, Rome, Italy, September 5-6, 2014. Revised Selected Papers*, volume 8902 of *Lecture Notes in Computer Science*, pages 128–143. Springer, 2014.

[Hir13] Tom Hirschowitz. Full abstraction for fair testing in CCS. In Reiko Heckel and Stefan Milius, editors, *Algebra and Coalgebra in Computer Science - 5th International Conference, CALCO 2013, Warsaw, Poland, September 3-6, 2013. Proceedings*, volume 8089 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2013.

[Hir14] Tom Hirschowitz. Full abstraction for fair testing in CCS (expanded version). *Logical Methods in Computer Science*, 10(4), 2014.

[HM99] Russell Harmer and Guy McCusker. A fully abstract game semantics for finite nondeterminism. In *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, pages 422–430. IEEE Computer Society, 1999.

[HO00] J. M. E. Hyland and C.-H. Ong. On full abstraction for PCF. *Information and Computation*, 163:285–408, 2000.

[HY97] Kohei Honda and Nobuko Yoshida. Game theoretic analysis of call-by-value computation. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 225–236. Springer, 1997.

[Joy77] André Joyal. Remarques sur la théorie des jeux à deux personnes. *Gazette des Sciences Mathematiques du Québec 1(4)*, pages 46 – 52, 1977.

[JPR12] Radha Jagadeesan, Gustavo Petri, and James Riely. Brookes is relaxed, almost! In *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012*, pages 180–194, 2012.

[KHL+17] Jeehoon Kang, Chung-Kil Hur, Ori Lahav, Viktor Vafeiadis, and Derek Dreyer. A promising semantics for relaxed-memory concurrency. In Giuseppe Castagna and Andrew D. Gordon, editors, *POPL 2017*, pages 175–189. ACM, 2017.

[Lai97] James Laird. Full abstraction for functional languages with control. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 58–67. IEEE Computer Society, 1997.

[Lai99] James David Laird. *A semantic analysis of control*. PhD thesis, University of Edinburgh, UK, 1999.

[Lai01] James Laird. A game semantics of idealized CSP. *Electr. Notes Theor. Comput. Sci.*, 45:232–257, 2001.

[Lam79] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.

[Loa01] Ralph Loader. Finitary PCF is not decidable. *Theor. Comput. Sci.*, 266(1-2):341–364, 2001.

[LS88] Joachim Lambek and Philip J Scott. *Introduction to higher-order categorical logic*, volume 7. Cambridge University Press, 1988.

[Mel03] Paul-André Melliès. Asynchronous games 1: A group-theoretic formulation of uniformity. *Manuscript, Available online*, 2003.

[Mel05a] P.A. Melliès. Asynchronous games 3 an innocent model of linear logic. *Electronic Notes in Theoretical Computer Science*, 122:171–192, 2005.

[Mel05b] Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA, Proceedings* **[DBL05]**, pages 386–395.

[Mel06] Paul-André Melliès. Asynchronous games 2: The true concurrency of innocence. *Theor. Comput. Sci.*, 358(2-3):200–228, 2006.

[Mil82] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[MM07] Paul-André Melliès and Samuel Mimram. Asynchronous games: Innocence without alternation. In *CONCUR 2007 - Concurrency Theory, 18th International Conference*, pages 395–411, 2007.

[MMS+12] Sela Mador-Haim, Luc Maranget, Susmit Sarkar, Kayvan Memarian, Jade Alglave, Scott Owens, Rajeev Alur, Milo M. K. Martin, Peter Sewell, and Derek Williams. An axiomatic memory model for POWER multiprocessors. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, pages 495–512, 2012.

[NSS+] Francesco Zappa Nardelli, Peter Sewell, Susmit Sarkar, Scott Owens, Luc Maranget, Mark Batty, and Jade Alglave. Relaxed memory models must be rigorous.

[OSS09] Scott Owens, Susmit Sarkar, and Peter Sewell. A better x86 memory model: x86-tso. In *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, pages 391–407, 2009.

[Plo77] Gordon D. Plotkin. Lcf considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.

[PP02] Gordon D. Plotkin and John Power. Notions of computation determine monads. In Mogens Nielsen and Uffe Engberg, editors, *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*, volume 2303 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2002.

[RW11] Silvain Rideau and Glynn Winskel. Concurrent strategies. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 409–418, 2011.

[SSA+11] Susmit Sarkar, Peter Sewell, Jade Alglave, Luc Maranget, and Derek Williams. Understanding POWER multiprocessors. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2011, San Jose, CA, USA, June 4-8, 2011*, pages 175–186, 2011.

[TO15] Takeshi Tsukada and C.-H. Luke Ong. Nondeterminism in game semantics via sheaves. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 220–231. IEEE Computer Society, 2015.

[vGP09] Rob J. van Glabbeek and Gordon D. Plotkin. Configuration structures, event structures and petri nets. *CoRR*, abs/0912.4023, 2009.

[Win82] Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Automata, Languages and Programming, 9th Colloquium, Aarhus, Denmark, July 12-16, 1982, Proceedings*, volume 140 of *Lecture Notes in Computer Science*, pages 561–576. Springer, 1982.

[Win86] Glynn Winskel. Event structures. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, 8.-19. September 1986*, pages 325–392, 1986.

[Win07] G. Winskel. Event structures with symmetry. *Electronic Notes in Theoretical Computer Science*, 172:611–652, 2007.

[Win12] Glynn Winskel. Deterministic concurrent strategies. *Formal Asp. Comput.*, 24(4-6):647–660, 2012.